E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Characterization of optical aberrations: Practical Implementation of a SHACK-HARTMANN Sensor with MATLAB

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Iñigo Belío Apaolaza

Lukas Traxler (FH Technikum Wien)

Luis Serrano Arriezu (UPNA)

Viena, 30 de junio de 2019

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Abstract

In cataract surgery, the clouded natural lens is replaced by an artificial lens called intra ocular lens (IOL). These lenses need to be correctly characterized for a successful surgery outcome, in order to avoid or compensate optical aberrations in the human vision system. In this thesis, an implementation of Shack-Hartmann wavefront reconstruction and characterization system is developed in MATLAB, with the objective to improve the performance of the standard software of a commercial sensor. Multiple approaches were implemented, focusing on optimizing correlation-based centroiding. Simulations were carried out to test the performance of different technique for noise cancelling and range extension, with a simple simulation environment developed in this thesis. Results show success on optimizing correlation-based centroiding, representing the closest technique to optimum in the majority of scenarios, and highlight the big impact that domain of reconstruction has in the system results. In conclusion, the system developed is flexible, implements a nice variety of options for adapting to different situations and provides the tools for testing and conducting research on the performance of Shack-Hartmann sensors. In addition, it is highly susceptible to modifications and improvements, and offers a solid alternative to the commercial sensor.

Keywords: Shack-Hartmann, Wavefront sensing, IOL, Centroiding methods, Correlation centroiding

Kurzfassung

Bei der Kataraktoperation wird die getrübte natürliche Linse durch eine künstliche Linse ersetzt, die als Intraokularlinse (IOL) bezeichnet wird. Diese Linsen müssen für ein erfolgreiches Operationsergebnis korrekt charakterisiert/kategorisiert werden, um optische Aberration beim menschlichen Sehen zu vermeiden oder zu kompensieren.In dieser Arbeit wird eine Implementierung des Shack-Hartmann Wellenfront-Rekonstruktions- und Charakterisierungssystems in MATLAB entwickelt, um die Leistung der Standardsoftware eines kommerziellen Sensors zu verbessern. Es wurden mehrere Ansätze implementiert, die sich auf die Optimierung des Korrelations-basierten Zentrierung konzentrieren. Es wurden Simulationen durchgeführt, um die Leistung verschiedener Techniken zur Geräuschunterdrückung und Range-Erweiterung mit einer einfachen Simulationsumgebung zu testen, die in dieser Arbeit entwickelt wurde.Die Ergebnisse zeigen den Erfolg bei der Optimierung des korrelationsbasierten Zentrieren, der in den meisten Szenarien die dem Optimum am nächsten kommende Technik darstellt, und verdeutlichen den großen Einfluss, den die Domäne der Rekonstruktion auf die System Ergebnisse hat. Zusammenfassend lässt sich sagen, dass das entwickelte System flexibel ist, eine Reihe von Anpassungsmöglichkeiten für unterschiedliche Situationen bietet und die Werkzeuge für Tests und Leitungsforschung zur Leistung von Shack-Hartmann Sensoren bereitstellt. Darüber hinaus ist es sehr aufnahmefähig für Änderungen und Verbesserungen und bietet eine solide Alternative zum kommerziellen Sensor.

Schlagwörter: Shack-Hartmann, Wavefront sensing, IOL, Centroiding methods, Correlation centroiding

Declaration of Authenticity

"As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (for example see §§ 21, 46 and 57 UrhG (Austrian copyright law) as amended as well as § 11 of the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

In particular I declare that I have made use of third-party content correctly, regardless what form it may have, and I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see § 11 para. 1 Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool."

Vienna, 30-06-2019

Place, Date

Signature

Table of contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AO	Adaptive Optics
OA	Optical Aberrations
IOL	Intraocular Lenses
SH	Shack-Hartmann
OPL	Optical Path Length
MA	Monochromatic Aberration
CA	Chromatic Aberration
SA	Spherical Aberration
CoG	Center of Gravity
WCoG	Weighted Center of Gravity
SNR	Signal to Noise Ratio
BMP	Best Matching Pixel
COD	Complete Orthogonal Decomposition
SVD	Singular Value Decomposition
RMSE	Root Mean Square Error

Acknowledgments

I would first like to thank my supervisor Dr. L. Traxler, whose expertise, support and orientation made this bachelor's thesis possible.

I would like to thank Dr. L. Serrano, for being very supportive and always available to solve problems and making this stay possible. Special thanks to Dr. Drauschke, for reviewing my writing and solving any theoretical issues I encountered throughout the project.

To my colleagues at UPNA and FH Technikum Wien: I'm very grateful of all the time we spent together. Thank you for accompanying me throughout the degree. Thank you for those endless days of studying before the exams and all the good memories I take. I am a better engineer and person because of you.

Finally, thanks to my family. To my mother, Ana, to my father, Javier, and to my sister, Helena. Thank you for supporting me no matter what, for inculcating the values I am proud of and the importance of hard work. You are the cornerstone of my life. I am what I am thanks to you.

Chapter 1

Introduction

In this first chapter, the motivation and objectives of the thesis are presented as well as the state-of-the-art of this thesis topic. Finally, the structure of this document is explained to guide the reader.

1.1 Motivation and state-of-the-art

Cataract is a clouding of the eye lens [1]. This condition is the "*main cause of blindness in middle and low-income countries*" and it is the "*second leading cause of visual impairment after refractive errors*" [2]. As reported in [2], there are over 24 million Americans older than 40 affected by cataracts. In the EU, only in 2016, cataract surgery was conducted 4.5 million times [3].

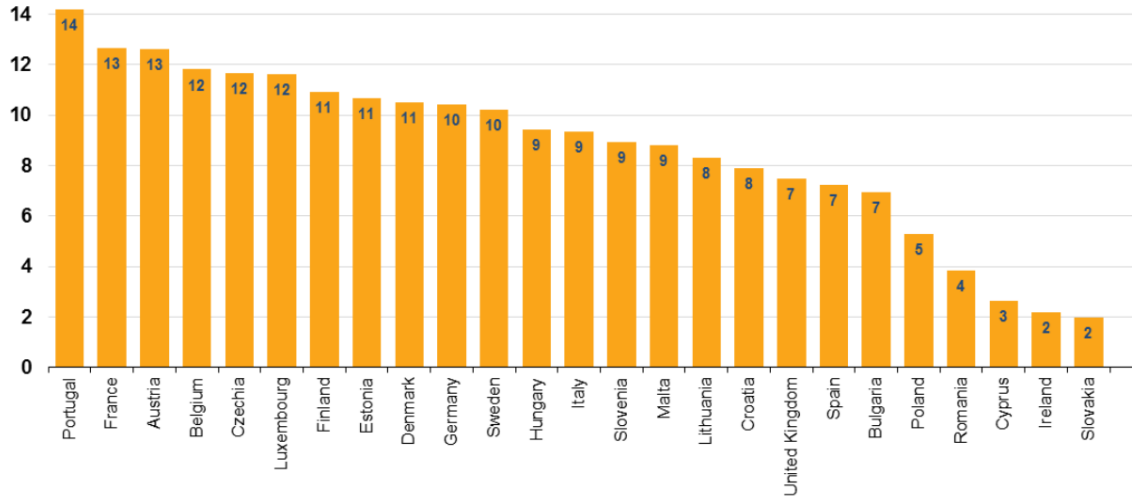


Fig-1: *Number of operations per thousand inhabitants among European countries. Data from 2016, except Portugal and Spain, which data comes from 2015. [3]*

Cataract surgery consists on a replacement of the clouded natural lens, by an artificial lens, called intraocular lens (IOL) [1]. In this sense, the characterization of the IOL is crucial for a successful surgery outcome. There is a need for correctly characterize optical aberrations present in the IOL.

The historical context of optical aberrations characterization remotes to the 20th century. Historically, the discipline that drove the advances on this field was Adaptive Optics (AO). We call AO to the "*scientific and engineering discipline whereby the performance of an optical signal is improved by using information about the environment through which it passes*" [4]. The basic goal of AO is stated as: "*to measure the aberrations of an incoming wavefront and then cancel these out by applying compensating aberrations*" [5]. The first report of the use of this discipline corresponds to Babcock 1953 [4], [6], [7] when he proposed the use of optical deformable elements, driven by a wavefront sensor, in order to compensate for "*distortions caused by the atmosphere*" [4] that affected images taken by telescopes. In the last decades, the use of AO has extended to very diverse fields, such as vision science, microscopy, microelectronics, lasers and physics, defense and space or Free-space optical communication [8]. In the medical context, it is remarkable the use of AO in retinal imaging and ophthalmology. The emergence of the use in ophthalmology dates from 1996 (Rochester) [6] and was driven by the development of SH sensor, in which this thesis focuses.

This context of multiple applications with high demand have led to a notorious development of AO techniques in the last decades, and therefore significant advances on characterization of OA techniques. The characterization of OA starts with the sensing and reconstruction of the wavefront. A physical description of the concept of wavefront can be found in section 2.2. In order to characterize wavefront aberrations, we always use indirect measurements from intensity distribution patterns on CCD or CMOS screens. Many sensing solutions have been proposed, in [7] a comprehensive review can be found concerning the different techniques developed. Among others, we find SH wavefront sensor, Curvature Wavefront Sensing, Wavefront Shearing Interferometry and Pyramid Sensors. Figure 2 depicts the different sensing technologies that have been developed since last decades.

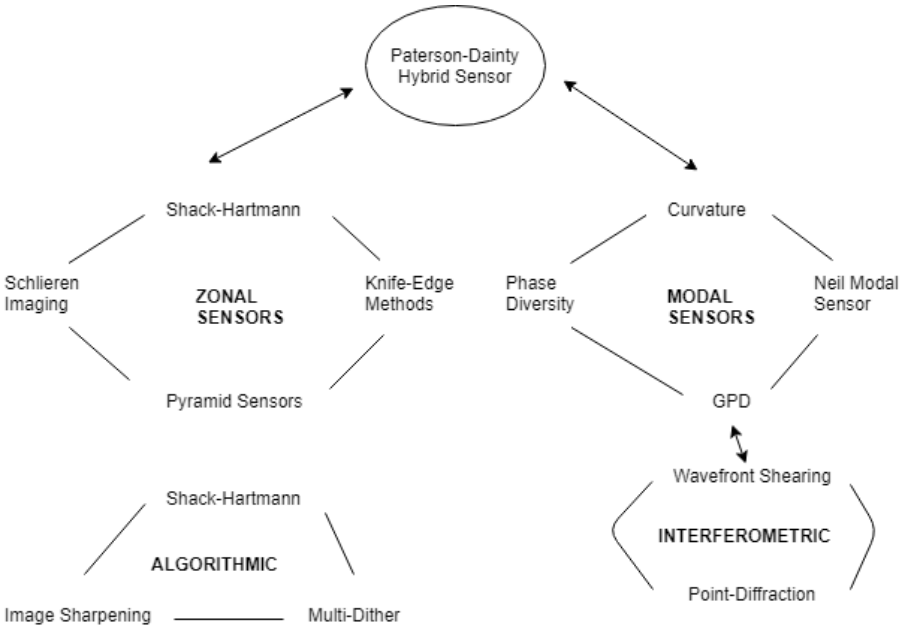


Fig-2: *Summary of the wavefront sensing techniques. Arrows illustrates link between the sensors. [5]*

The use of each kind of sensor depends highly on the application. However, the SH sensor's white light capability and optical efficiency make it a favorite among astronomers, but also the sensor is used extensively in medical applications. This work focuses exclusively on SH sensor's.

Wavefront reconstruction is still an opened research topic with many different approaches. One of the most relevant and early contributions to wavefront reconstruction algorithms corresponds to Southwell [9], where he presented zonal and modal estimation techniques. Many subsequent research has been done since then. Authors have improved these algorithms, proposed new basis functions for modal estimation and presented noise cancelling and range extension techniques. This work gives an insight into the classical wavefront reconstruction techniques and explores various approaches proposed by authors in order to improve the characterization of OA from SH sensor measurements.

1.2 Aim and objectives

There are multiple commercial SH sensors available in the market. Although they incorporate a software suite to measure wavefront and aberrations, they are often limited and static. In this thesis we work with the sensor WFS-150-7AR from ThorLabs [10]. The aim of this thesis is to implement the software of the wavefront reconstruction and decomposition to measure aberrations in IOL, in order to provide a flexible environment as an alternative to the standard software of the commercial sensor. In this sense, multiple techniques presented by authors will be studied and implemented to enhance the performance of the system. The objectives of the thesis can be summarized as:

- **Implementing a basic wavefront reconstruction and decomposition system of a SH sensor:** Starting from raw data captured by the camera of the sensor a full functioning software system of reconstruction and characterization of aberrations will be developed. The chosen platform to perform this objective is MATLAB.
- **Improving the basic system by implementing and testing noise reduction techniques and increasing the dynamic range:** As will be discussed later, there are multiple paths to implement the system. Different techniques for noise reduction and wavefront reconstruction will be implemented to enhance the performance of the system, but also to give the user flexibility when using the software.
- **Optimizing correlation-based wavefront sensing:** A new simple technique is presented to optimize correlation-based sensing by an adaptive template. The technique will be developed and tested it in a simulation environment also elaborated in this thesis.

1.3 Structure of the thesis

This work consists of 7 chapters. After this introduction, chapter 2 will cover the theoretical background of ray optics, wavefront and optical aberrations. In chapter 3, the principles of the sensor will be discussed, as well as diverse reconstruction approaches. In chapter 4, we explore techniques that can be performed for noise reduction and range extension. If the reader is familiarized with the background of wavefront sensing and reconstruction, he might jump this chapter. Chapter 5 covers the software implementation that has been conducted with MATLAB, describing every part of the program. Chapter 6 corresponds to the results and discussion. Finally, in Chapter 7 the conclusion is found.

Chapter 2

Physical Background

In this chapter the necessary theoretical background is provided. First, the approach of ideal optical system is presented based on ray optics. The concept of wavefront is described next, giving both a physical and geometric interpretation. Then aberrations on an imaging system are discussed, focusing on geometric errors, and explaining how these aberrations are related to the Zernike Polynomials.

2.1 Ideal optical system

Light is one more manifestation of the electromagnetic phenomena. However, there exists different models for describing light, as we try to explain more complex light interaction with the environment. Basically, there are ray optics, wave optics, electromagnetics optics and quantum optics [11].

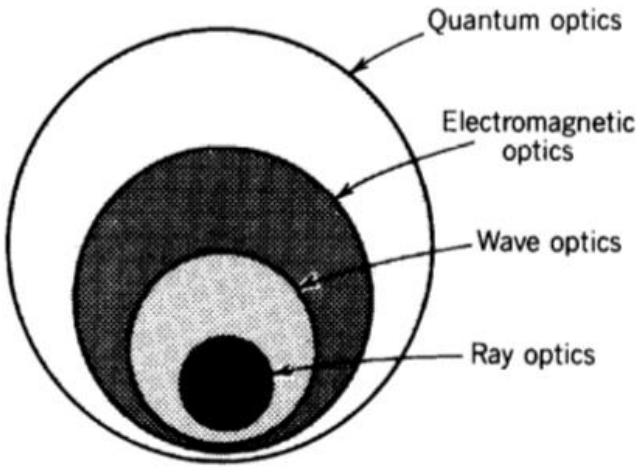


Fig-3: *Theory of light. "The theory of quantum optics provides an explanation of virtually all optical phenomena. The electromagnetic theory of light provides the most complete treatment of light within the confined of classical optics. Wave optics is a scalar approximation of electromagnetic optics. Ray optics is the limit of wave optics when the wavelength is very short." [11]*

The lowest approximation level theory to describe light interaction is ray optics. For most of the physical phenomena concerning this thesis, this level is sufficient to describe them. Ray optics gives us information about the direction of light rays, and, therefore, with this approximation image formation can be studied. The ray theory of light emerges from 4 simple postulates, stated in [11]:

- Light propagates in the form of rays. *"Rays emitted by light sources can be observed when reaching an optical detector"* [11].
- Every optical medium can be characterized by its refractive index n. The refractive index is the ratio between the speed of light in free space c_0 and the speed in the medium c, so that $n \geq 1$. Therefore, the time taken by light to travel a distance d equals $d/c = nd/c_0$. The product nd is known as the optical path length (OPL).
- In a not homogeneous medium, the refractive index $n(r)$ is a function of the position $r = (x, y, z)$. The OPL along any given path between two points A and B is therefore,

$$OPL = \int_A^B n(r)ds \tag{2.1}$$

- The 4th postulate is known as the Fermat's Principle. It basically states that the OPL travelled by optical rays between two points, A and B, will be always the minimum possible. In other words, light rays travel along the path of least time.

With these postulates all the ray optics laws can be derived and reflection, propagation, refraction and of light can be explained. When talking about an optical system, the optical axis is identified, which is the axis about the optical system is centered. Around this axis, there is a set of important rays that travel at small angles (less than 7^0 typically). Such rays are called paraxial rays. Usually, we can assume that in our system only have paraxial rays are present; this is the basic assumption for paraxial optics. To describe the variation of inclination and position of a paraxial ray propagating across an optical system, a 2x2 matrix can be used. This method is known as matrix optics.

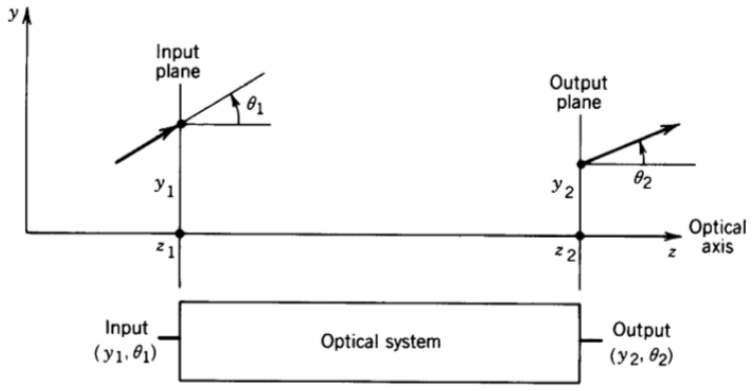


Fig-4. *Matrix optics scheme. [11]*

In the paraxial approximation, when all the angles are sufficiently small ($\sin\theta \approx \theta$), the relation between (y_2, θ_2) and (y_1, θ_1) can be written as,

$$y_2 = Ay_1 + B\theta_1 \tag{2.2}$$

$$\theta_2 = Cy_1 + D\theta_1 \tag{2.3}$$

where A, B, C and D are real numbers. Equations 2.2 and 2.3 can be written in their matrix form as,

$$\begin{bmatrix} y_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} y_1 \\ \theta_1 \end{bmatrix} \tag{2.4}$$

The matrix method allows as to characterize any optical system if we know the parameters A, B, C and D, since it permits (y_2, θ_2) to be determined for any (y_1, θ_1). It is known as the ray-transfer matrix.

With this method we can describe easily any optical system taking the paraxial approximation. Some examples are shown in figure 5:

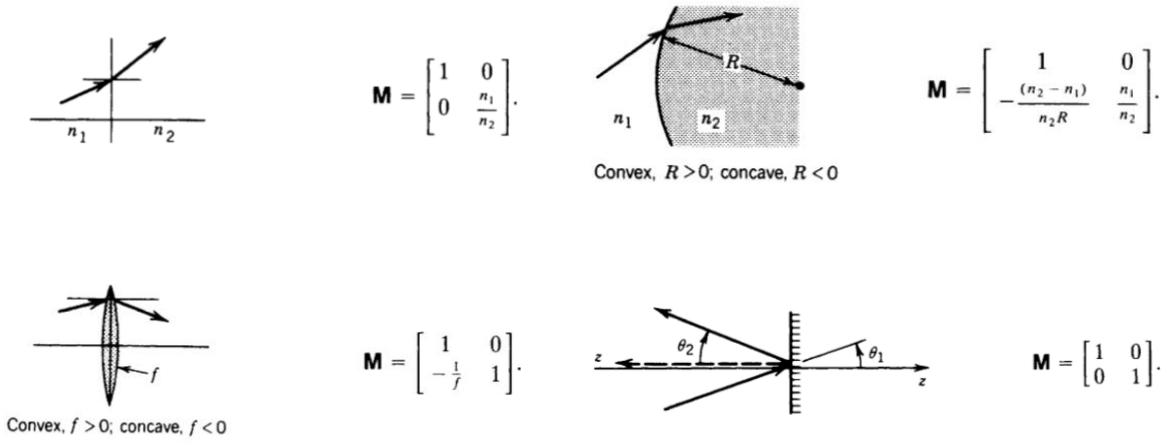


Fig-5: *Examples of ray-transfer matrix of different optical systems. From left to right and top to bottom: refraction at a planar boundary, refraction at a spherical boundary, transmission through a thin lens, reflection from a planar mirror. [11]*

That said, an ideal optical system is the one where light emitted from a point P in any angle will converge into a single point P' after passing through the system. In the matrix form, we can see this by looking at the element B. This element relates how the output height depends on the input ray angle; if the element is 0 that means that all the rays emitted from P, will eventually converge into a single point P'.

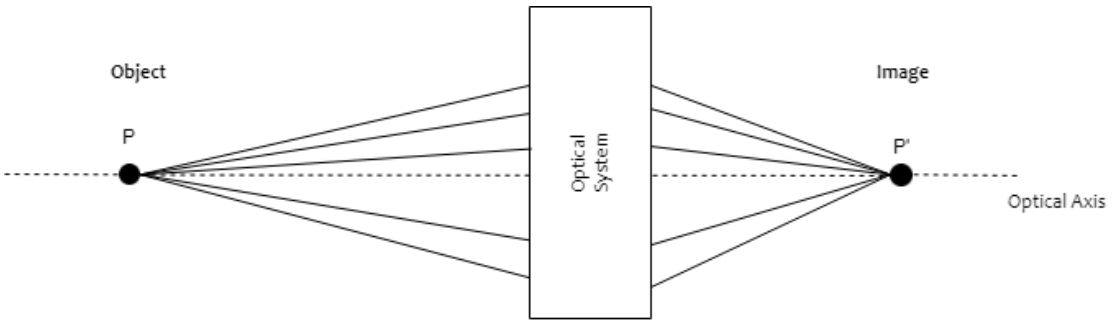


Fig- 6: *Ideal Imaging System.*

Any system that doesn't verify this imaging condition will present some kind of optical aberration. This will be discussed in detail in 2.3.

2.2 Wavefront

In order to understand the concept of wavefront, wave nature of light must be considered. Recalling figure 2, the theory that describes light as a wave is known as wave optics. We will not describe in detail the fundaments of the theory, as ray optics is enough to describe the phenomena of interest of this thesis, but to comprehend the wavefront a few definitions must be done.

An optical wave is "*described mathematically by a real function*" [11] of position $r = (x, y, z)$ and time t, written as $u(r, t)$ and known as the wavefunction. Every wavefunction must fulfill the wave equation:

$$\nabla^2 u - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = 0 \tag{2.6}$$

A common solution for the equation is the monochromatic wave, which have harmonic time dependency:

$$u(r, t) = a(r)\cos[2\pi f t + \psi(r)] \tag{2.7}$$

where $a(r)$ is the amplitude, $\psi(r)$ is the phase and f the frequency.

Usually we represent the real wavefunction in terms of a complex function:

$$U(r,t) = a(r)e^{j(2\pi ft+\psi(r))} \tag{2.8}$$

so that:

$$u(r,t) = Re\{U(r,t)\} \tag{2.9}$$

The function $U(r,t)$ is called the complex wavefunction and describes the wave completely. The wave equation must be also satisfied by the complex wavefunction:

$$\nabla^2 U - \frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} = 0 \tag{2.10}$$

The most basic solutions for the wave equation are the plane and spherical wave. The complex wavefunction for each of these waves is the following:

PLANE WAVE

$$U(r,t) = Ae^{(-jkr)}e^{(-j2\pi ft)} \tag{2.11}$$

SPHERICAL WAVE

$$U(r,t) = \frac{A}{r}e^{(-jkr)}e^{(-j2\pi ft)} \tag{2.12}$$

where A is a constant known as the complex envelope. $k = (kx, ky, kz)$ is the wavevector and must satisfy $k^2 = kx^2 + ky^2 + kz^2$. The direction of propagation coincides with the direction of k.

That said, the concept of wavefront can be easily stated. Basically, the wavefronts are surfaces of the wave with equal phase, $\psi(r) = constant$. The constant is often taken to be multiples of 2π, $\psi(r) = 2\pi q$, where q is an integer. The maximum rate of phase change is indicated by the wavefront. In other words, we can see the wavefront as: *"an imaginary surface joining all points in space that are reached at the same instant by a wave propagating through a medium" [11].*

From equations (2.11) and (2.12) we can obtain the expression for the wavefront. In case of the plane wave the phase is $\arg\{U(r)\} = \arg\{A\} - kr$, thus the wavefronts are perpendicular planes to k.

To interpret the concept of wavefront with ray optics, the relation between wave optics and ray optics must be stated. Basically, the rays are pointing in the direction of propagation of the wave, thus the wavefront is perpendicular to the light rays. In this way, by tracking the direction of the rays the shape of the wavefront can be obtained. A geometric interpretation is shown in figure 7:

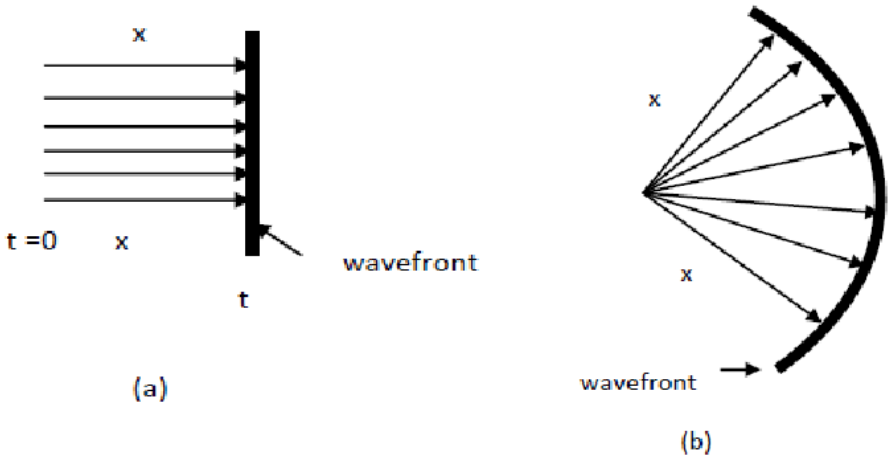


Fig- 7. *Geometric Interpretation of the wavefront (a): plane WF from a plane wave; (b) spherical WF from a spherical wave. [12]*

Wavefront analysis is a powerful tool, because it gives us information about the characteristics of the optical system. Taking the example of a converging lens, we can easily see how the lens modifies the incoming plane wavefront to a spherical (figure 8).

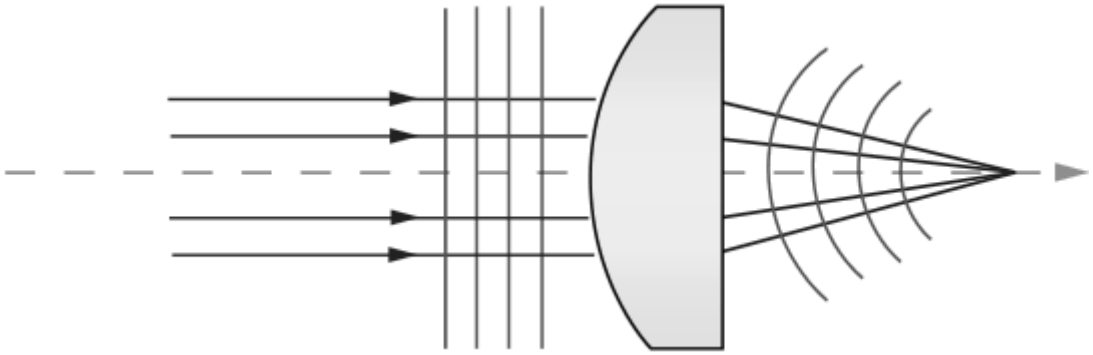


Fig- 8. *An incident plane wavefront goes through a converging lens (plano-convex), and the resulting wavefront is spherical. [13]*

A comprehensive summary of this section is represented by the following figure:

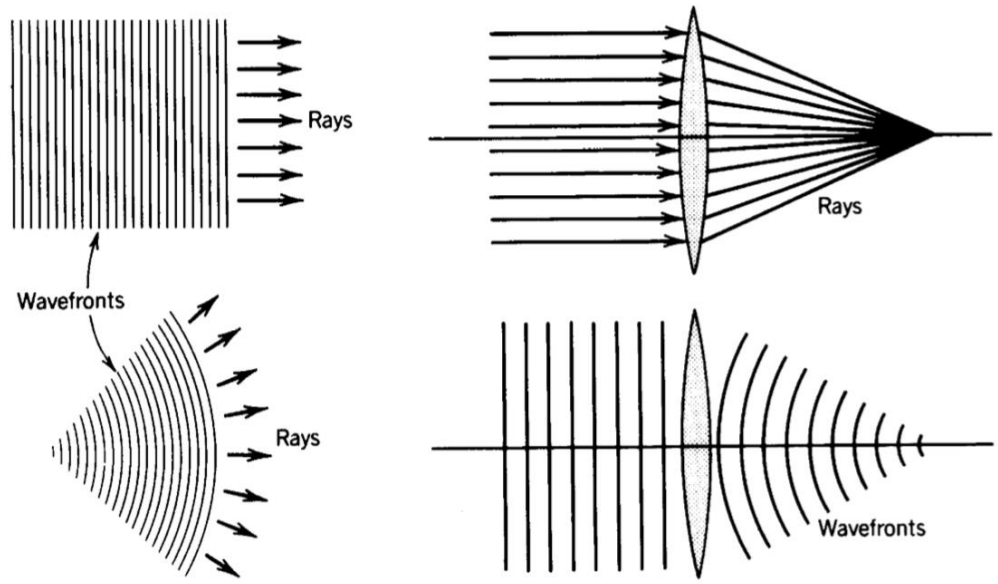


Fig- 9. *Relation between wavefront, ray optics and wave optics and the effect of a lens on rays and wavefronts. [11]*

2.3 Optical Aberrations

Recalling 2.1, for an optical system to be ideal, all rays emitted from a point P must converge into a single point P' after going through the system. Any optical system that doesn't verify this condition presents some optical aberration.

In general terms, aberrations are image defects that arise from characteristics of the spherical surfaces used in the optical system. These image imperfections are generated by the spherical surfaces themselves, which produce the reflection or refraction of light, and are not necessarily the results of poor fabrication techniques, material properties, or, mounting techniques [14].

There are different classifications for optical aberrations. Typically, we distinguish between monochromatic aberrations (MA) and chromatic aberrations (CA) [14]. MA are referred to the aberrations that appear within the same wavelength, whereas CA occurs due to the different behavior of the optical system to different wavelengths. The first ones are also referred as geometric aberrations, because they infringe the geometric condition for the optical system discussed in 2.1. In this work the study is limited to MA aberrations, thus CA won't be discussed. Down below, the most common MA are briefly explained, that are: defocus, spherical aberration, coma, astigmatism and trefoil. The definitions of the geometric errors discussed, are taken from [1], [4], [15], [16], [17].

In order to discuss the geometric errors, we will assume that the object is situated in infinity. As a result, light rays enter parallel to the optical system. As [1] points out, *''this assumption is also used in ophthalmology, since the relaxed eye is accommodated to infinity''*.

In order to give a picture of the effect of spherical aberrations on the wavefront we apply the reverse the ray path [1]. Recalling figure 8, when parallel rays enter an ideal optical system, they converge into a single point. The reverse effect will occur if the light is emitted from that point, i.e. rays coming out from the lens will be parallel and so will be the resulting wavefront. Under this assumption the plane WF from an ideal system can be compared to the deformed one.

DEFOCUS

Defocus is the simplest optical aberration. It can be interpreted *"as the translation of the focus along the optical axis away from the detection surface"* [17]. This misalignment produces a blurred image. Although it violates the geometric condition for the ideal system, it is often not classified as an optical aberration, because it is a matter of adjusting the detection surface. However, in the human eye we find focusing issues very frequently. When the eye is captures the image too close (before the rays converge) it is called Hyperopia and when the eye captures it too far, it is called Myopia.

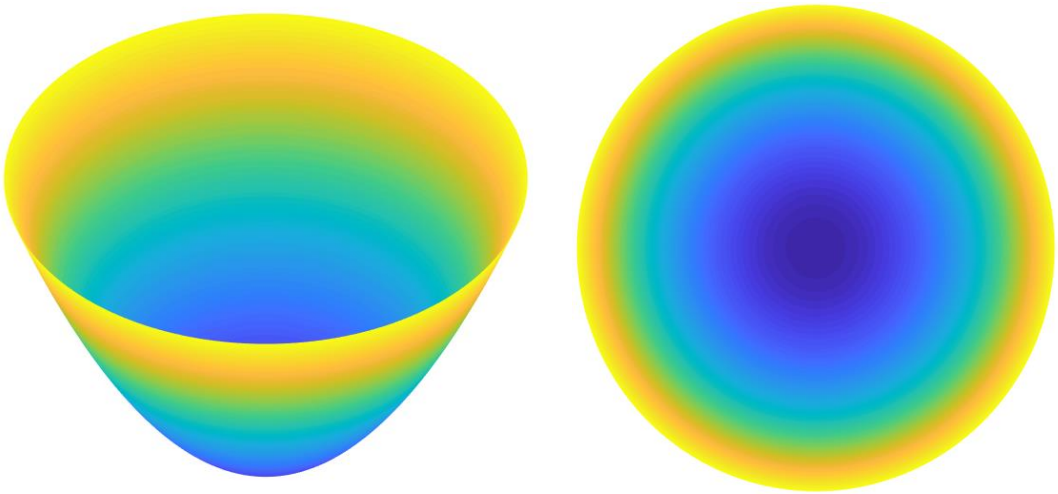


Fig-10: *Wavefront shape of a defocus. The representation depicts a gradual phase change from center to edges. On the left, a 3D view is shown. On the right, a vertical view from above.*

SPHERICAL ABERRATION

Spherical aberration (SA) is an inherent aberration that emerges from spherical surfaces. On a spherical surface, the focal length depends on the distance of the ray to the optical axis. As a result, rays located near the outter borders will be refleted or refracted with a different angle.In the human eye, SA is a very common condition, specially present in low low light conditions, where the pupil opens wider to receive more light, i.e admiting a larger range of incoming lights.

The SA produces a very characteristic ray envelope called the caustic curve, that is depicted in the following figure:

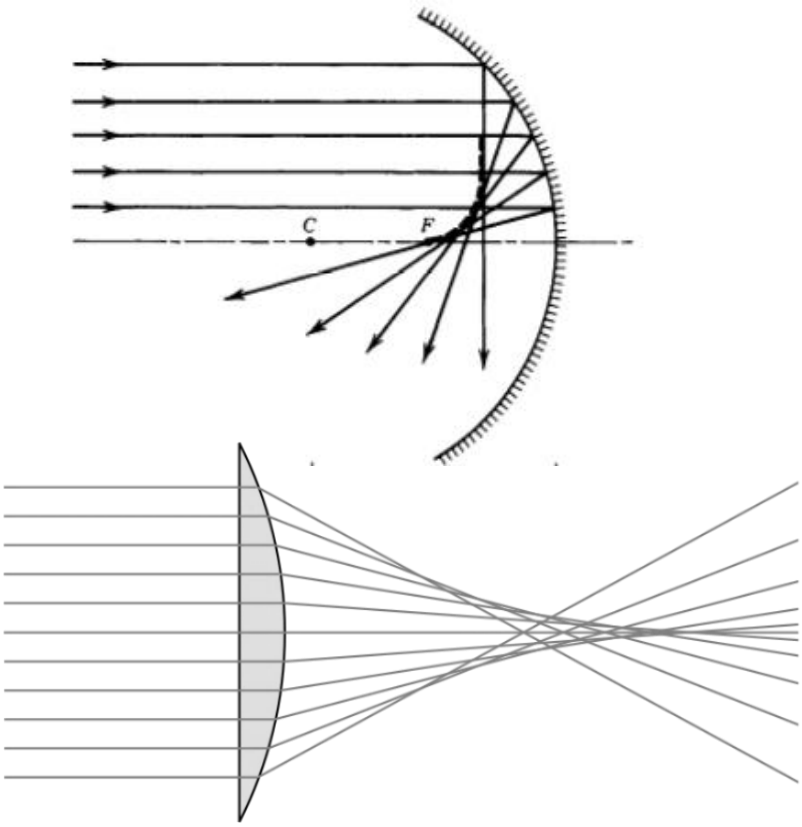


Fig-12: *Top: Reflection of parallel rays on a concave spherical mirror. The caustic curve is depicted as the dash line. Bottom: Spherical aberration effect on a lens. [11]*

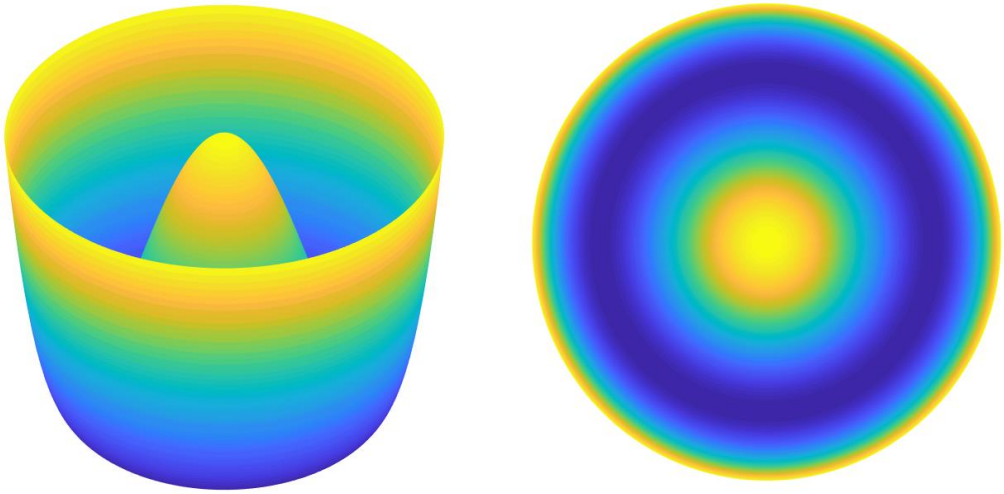


Fig- 11: *The wavefront resulting from spherical aberration have a sombrero-like shape.*

COMA

Coma aberration is similar to spherical; it applies to rays entering the lens at an angle. It can be defined as "*the variation of magnification with aperture*" [17]. Thus, paralell rays enter with some angle to the optical system, will be imaged at different heights, deppending on where the rays are entering the system. This produces a blurred point with a comet like tail, and it is named after this.

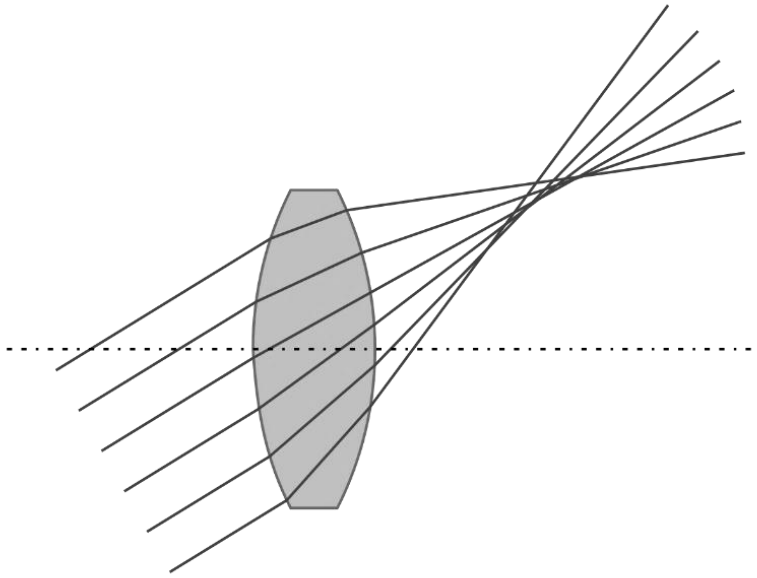


Fig-13: *Representation of coma aberration on a thick lens. Rays are entering the lens with same angle but are not forming image at the same height. [15]*

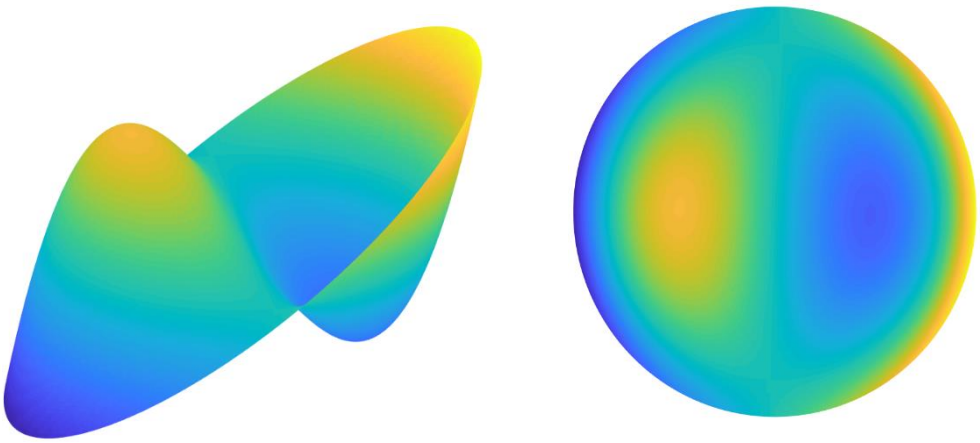


Fig -14: *Wavefront shape from coma aberration.*

ASTIGMATISM

Astigmatism is a word often used to describe a defect of the human eye. However In the formal optical sense, astigmatism refers to producing two images at two spatial locations. Another way to see it is as a different focal length of an optical system between two orthogonal axis.

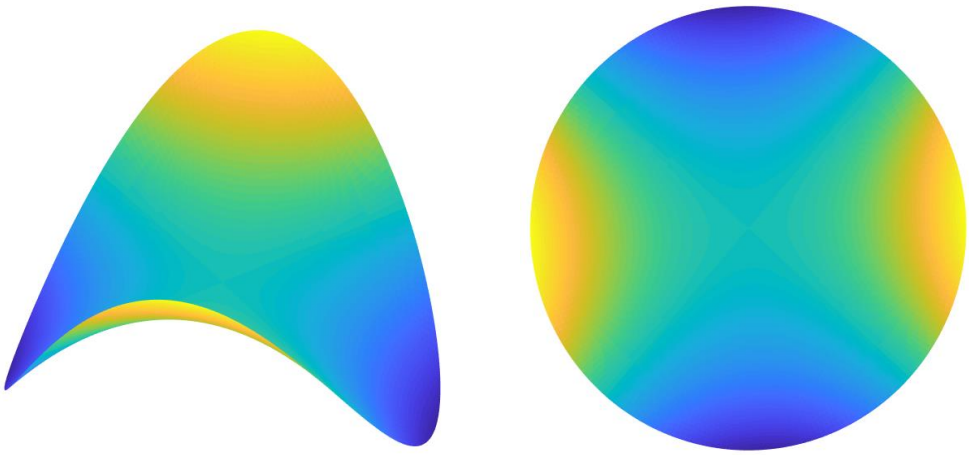


Fig- 15: *Wavefront shape resulting from astigmatism. The figure depicts how the phase distribution is opposed along the two main orthogonal axes.*

TREFOIL

Trefoil is another high order aberration. As well as for coma, and spherical aberration, it can be understood as a *"irregular"* astigmatism. This aberration causes light to smear in 3 directions, like Mercedes-Benz logo. Trefoil, along with Coma and SA, is a common high order aberration in the human eye.

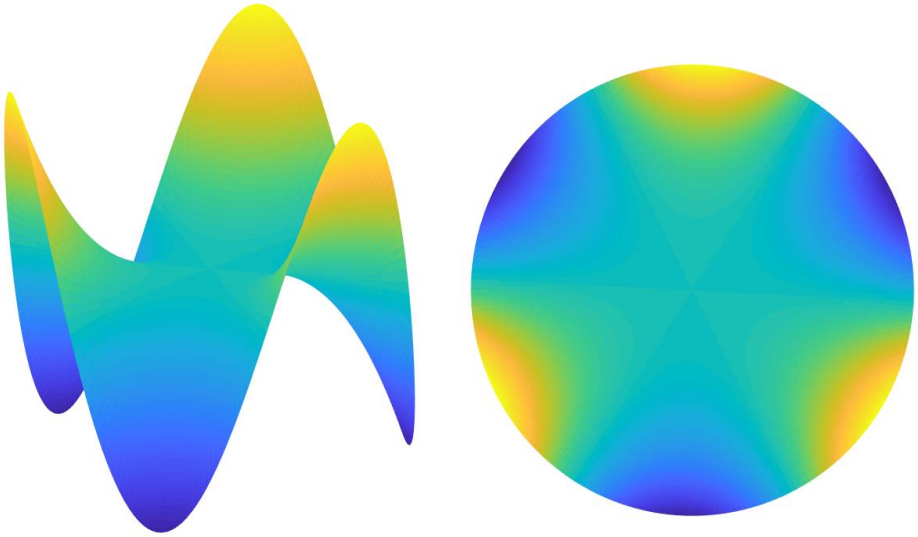


Figure 16: *Wavefront shape resulting of trefoil aberration. In the view from above (right), the characteristic three peak shape, like Mercedes-Benz Logo is appreciated.*

2.4 Zernike Polynomials

In general, the function that describes an arbitrary wavefront in polar coordinates (r, θ), denoted by $WF(r, \theta)$ can be expanded in terms of a set of orthogonal basis functions. There has been proposed multiple sets of basis functions like Karhunen-Loeve functions, Fourier series, Wavelets or Seidel Polynomials [18]. However, the most stablished functions in characterization of aberrations are the Zernike polynomials.

The Zernike polynomials are named after the physicist Frist Zernike, who won the Nobel prize in physics for the invention of phase-contrast microscopy in 1953 [19]. These set of polynomials are defined over the unit circle. The most important characteristic of the polynomials is the fact that each one represents a specific monochromatic aberration; thus, they are a powerful tool for wavefront analysis. Moreover, as they are orthogonal over the unit circle, they are ideal for describing wavefronts with circular shape. As a result, every optical system embedded on a circular pupil is highly suitable for Zernike expansion.

We can write the wavefront $WF(r, \theta)$ as:

$$WF(r,\theta) = \sum_{n,m} C_n^m Z_n^m(r,\theta) \tag{2.13}$$

where Z denotes the polynomials and C the amplitudes or coefficients. The Zernike polynomials written in polar coordinates ($x = rsin\theta\ ,\ y = rcos\theta$) are given by the following complex combination:

$$Z_n^m(r,\theta) \pm jZ_n^{-m}(r,\theta) = R_n^m(r)e^{\pm jm\theta} \tag{2.14}$$

which leads to

$$Z_n^m(r,\theta) = R_n^m(r)\cos m\theta \tag{2.15}$$

$$Z_n^{-m}(r,\theta) = R_n^m(r)\sin m\theta$$

where r is restricted to the unit circle $(0 \le r \le 1)$. $Z_n^m(r,\theta)$ corresponds to the even polynomials and $Z_n^{-m}(r,\theta)$ to the odd polynomials. The radial function, $R_n^m(r)$ is defined by:

$$R_n^m(r) = \sum_{l=0}^{(n-m)/2} \frac{(-1)^l (n-l)!}{l!\left[\frac{1}{2}(n+m)-l\right]!\left[\frac{1}{2}(n-m)-l\right]!} r^{n-2l} \tag{2.16}$$

We can rewrite equation 2.16 using products of binomials:

$$R_n^m(r) = \sum_{l=0}^{(n-m)/2} (-1)^l \binom{n-l}{l} \binom{n-2l}{\frac{n-m}{2}-l} r^{n-2l} \tag{2.17}$$

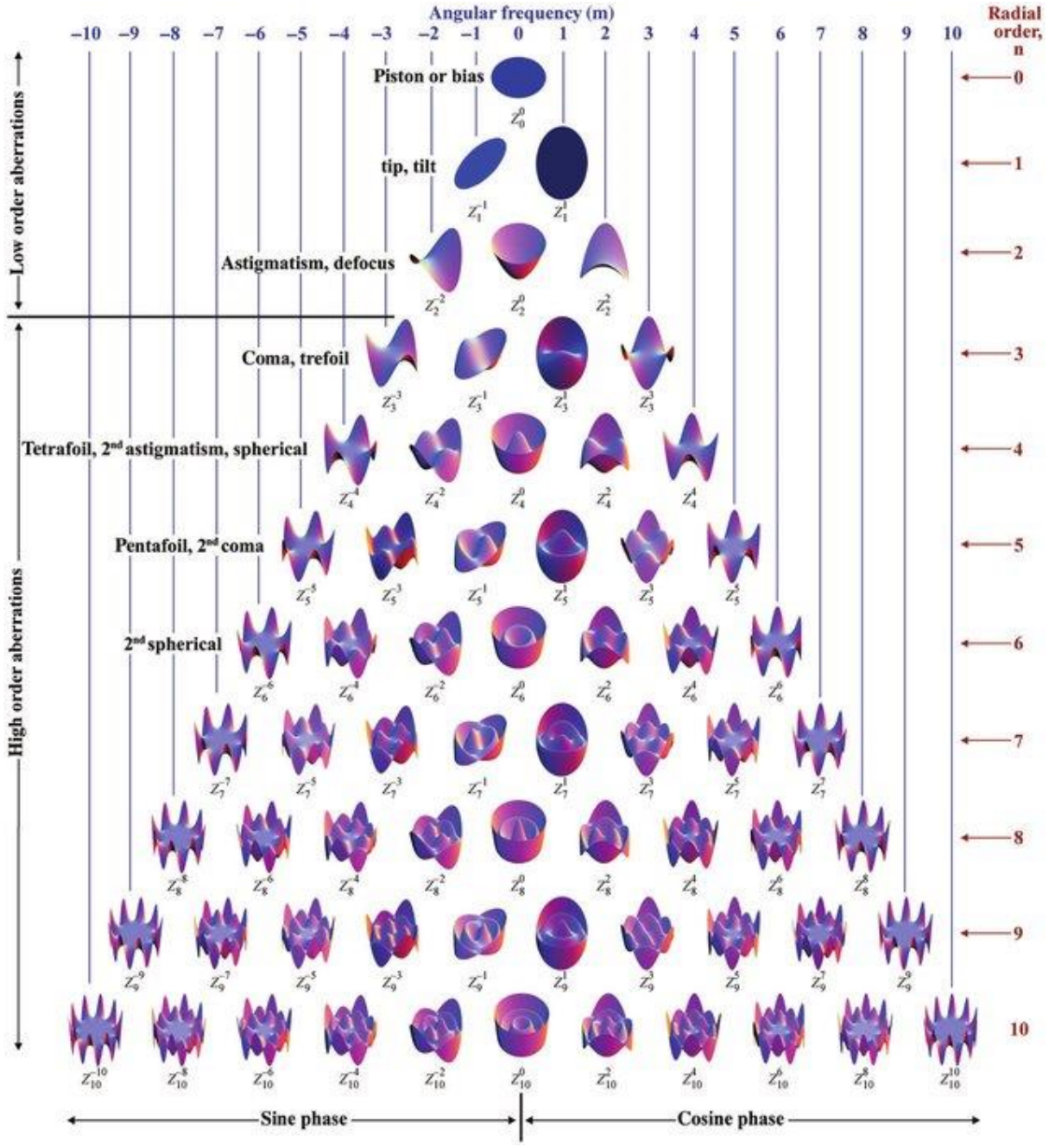


Fig-17: *3D plots of Zernike polynomials up to 10th order. The name of the classical aberration associated with some of them is also provided. [20]*

Typically, the double index *n* and *m* is used for describing Zernike polynomials. However, for convenience it is useful to define a single index *j*. This indexing strategy was introduced by Noll [20], although in this thesis, the indexing used corresponds to the given by OSA/ANSI [19]. The association of both indices systems is the following:

$$j = \frac{n(n+2)+m}{2} \tag{2.18}$$

n,m	j
0.0	0
1,-1	1
1,1	2
2,-2	3
2,0	4
2,2	5
3,-3	6
3,-1	7
3,1	8
3,3	9
4,-4	10
4,-2	11
4,0	12

Table 1: *Association of indices n and m to single index j.*

OSA/ANSI (j)	Radial degree (n)	Azimuthal degree (m)	Z_j	Aberration name
0	0	0	1	Piston
1	1	1	$2rcos\theta$	Tilt
2	1	-1	$2rsin\theta$	Tip
4	2	0	$\sqrt{3}(2r^2-1)$	Defocus
3	2	-2	$\sqrt{6}r^2sin2\theta$	Oblique Astigmatism
5	2	2	$\sqrt{6}r^2cos2\theta$	Vertical astigmatism
7	3	-1	$\sqrt{8}(3r^3-2r)sin\theta$	Vertica coma
8	3	1	$\sqrt{8}(3r^3-2r)cos\theta$	Horizontal coma
6	3	-3	$\sqrt{8}r^3sin3\theta$	Vertical trefoil
9	3	3	$\sqrt{8}r^3cos3\theta$	Oblique trefoil
12	4	0	$\sqrt{5}(6r^4-6r^2+1)$	Primary spherical
13	4	2	$\sqrt{10}(4r^4-3r^2)cos2\theta$	Vert sec astigmatism
11	4	-2	$\sqrt{10}(4r^4-3r^2)sin2\theta$	Obliq sec astigmatism
14	4	4	$\sqrt{10}r^4cos4\theta$	Vertical quadrafoil
10	4	-4	$\sqrt{10}r^4sin4\theta$	Oblique quadrafoil

Table 2: *First 15 Zernike Polynomials. In the third column, the mathematical expression is given, and, in the fourth column, the classical name given to the aberration.*

Chapter 3

Wavefront sensing background

In this chapter, the basic principles of operation of the Shack-Hartmann sensor are explained. We begin with an introduction of the sensor itself, its operating principle and characteristics, and then we move on to the wavefront reconstruction techniques, their singularities and issues.

3.1 Shack-Hartmann Sensor

The SH sensor technology started to be developed during the late 1960s by request of the US Airs Force [21] as a solution to improve images taken from earth by satellites. As an alternative to interferometry, the SH sensor is "*simple, compact, robust and relatively vibration insensitive*" [22]. It makes passive measurements of the incident light and it is wavelength independent. These advantages have led to formidable number of new applications in the last decades, although we find its historical application is found in the field of astronomy.

The Shack-Hartmann sensor is an evolution from the Hartmann test [5] which was used as a tool for determining the wavefront deformation caused by the shape of telescopes' mirrors. In this test, a mask containing a vast number of holes was placed over the primary mirror of the telescope and a CCD screen was placed in the focal plane. Using an emitting source, light was reflected at the mirror and then measured the displacements on the CCD sensor.

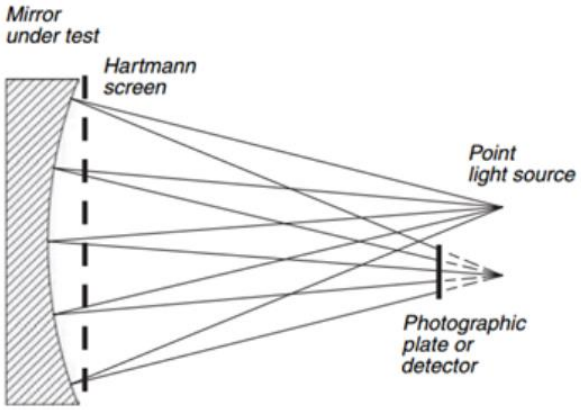


Fig- 18: *Scheme of Hartmann test for large telescopes. [23]*

The basic geometry of the SH sensor is depicted in figure 19. It consists on an array of micro-lenses placed a focal distance away from the detection screen, that is typically CMOS based. A light beam going through the array of lenses will produce a spot pattern on the screen, as every lens from the array focuses the light into a single dot on the screen. If the wavefront from the incident light beam is planar, then the pattern will be considered the reference. A distorted wavefront will produce a different dot pattern, and from comparing both patterns the distorted wavefront is eventually determined.

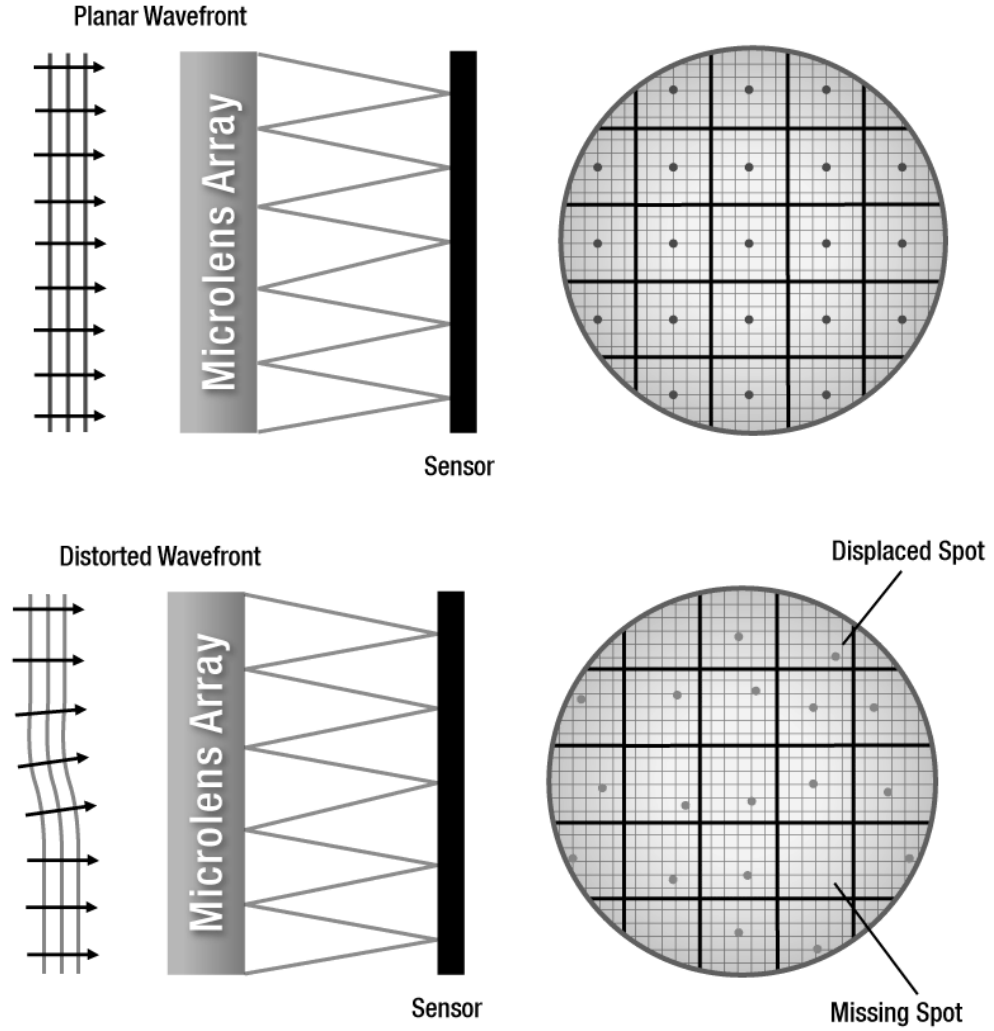


***Fig-19**: Basic geometry of a SH sensor. In the figure, we see how two different incoming wavefronts produce two different dot patterns. In some cases, where the distortion is too high, spots can leave their corresponding sub-areas, leading to wrong wavefront estimation. This can be fixed by applying range extension techniques, discussed in Chapter 4. [24]*

The sensor works under the assumption of locally plane wavefront. By assuming the wavefront to be plane at each micro-lens area, the displacement of the spot at the lens compared to the reference spot, is directly the gradient of the wavefront at that position, corresponding to the x and y direction.

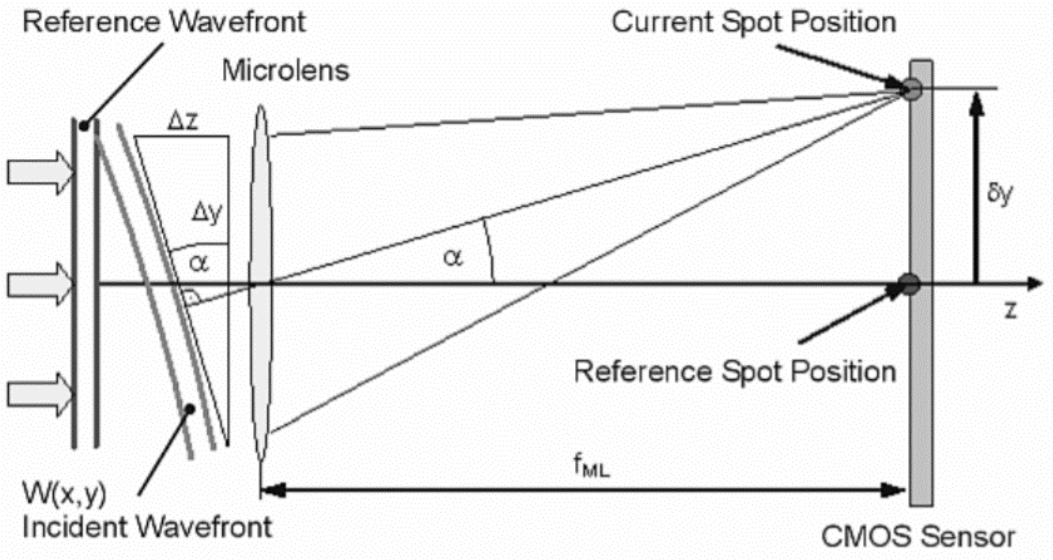


Fig-20: *Detailed view of distorted wavefront detection at a single micro-lens. [24]*

From figure 20, the parameters relevant to us are the angle of the distorted wavefront to the planar α, the diameter of the lens Δy, the distance between the micro-lens and the CMOS screen fml (which is typically the focal length of the lens), the separation between the reference spot and the displaced spot δx and δy, and the height of the distorted wavefront Δz. From the figure we obtain the following relationship:

$$tan\alpha = \frac{\Delta z}{\Delta y} = \frac{\delta y}{fml} \tag{3.1}$$

$W(x, y)$ describes the shape of the wavefront, and its partial derivatives relative to X and Y are determined by the spot shift δx and δy:

$$\frac{\partial}{\partial x} W(x,y) = \frac{\delta x}{fml} \;\; ; \;\; \frac{\partial}{\partial y} W(x,y) = \frac{\delta y}{fml} \tag{3.2}$$

That is the basic equation of a Shack-Hartmann sensor, and it shows that after a 2-dimensional integration process, the total wavefront can be calculated from the spot deviations. The full process of wavefront reconstruction will be discussed in the next section.

3.2 Wavefront reconstruction

In this section, classical wavefront reconstruction methods that will be implemented by software are discussed. There exist multiple strategies to follow, and the problem of wavefront reconstruction is still an opened research topic. In this sense, there is not an error free wavefront reconstruction. All methods are estimations for the wavefront shape from the slope measurements. These estimations are numerical solutions to the gradient equation [25]:

$$\nabla W = S^x i + S^y j \tag{3.3}$$

where W is the wavefront, S^x and S^y are the partial derivatives of the wavefront and *i* and *j* are basis unit vectors. This equation can be discretized over the grid corresponding to the micro-lens array that we showed in the previous section resulting in:

$$\nabla W = S^X_{n.m} i + S^Y_{n.m} j \tag{3.4}$$

where *n, m* corresponds to the index for each grid position. In the following sub-sections, the different numerical solutions to equation 3.4 will be explained, including their problems and challenges.

3.2.1 Displaced Spot Position and Slope measurements

The first step in measuring the wavefront from CMOS image coming from the Shack-Hartmann sensor is to determine the location of the displaced spots. The location is determined by the CoG (center of gravity) or centroid calculation. In general, for and arbitrary intensity pattern the centroid in the x direction is given by:

$$\bar{x} = \frac{\iint_{-\infty}^{\infty} I(x,y)\, x\, dxdy}{\iint_{-\infty}^{\infty} I(x,y)\, dxdy} \tag{3.5}$$

where $I(x, y)$ is the intensity function of x and y, and x is the location at which the intensity is measured along the x-axis. For the y direction the equation is identical. We can discretize this equation to apply on images from a CMOS screen. At each position of the grid, the discrete equation can be applied. Assuming a rectangular grid we can write equation 3.5 as:

$$\bar{x} = \frac{\sum_{i=i_{min}}^{imax} \sum_{j=j_{min}}^{jmax} I(i,j)i}{\sum_{i=i_{min}}^{imax} \sum_{j=j_{min}}^{jmax} I(i,j)} s \tag{3.6}$$

where $I(i,j)$ is the intensity measured in the i-row and j-column of the grid position and s ($\mu m/pixels$) is the spacing of pixels along the *x* or *y* axes. By doing this, we obtain the position of the centroid in units of μm.

This operation is performed for each grid position of the CMOS image, but also for the calibrated reference. This reference corresponds to the intensity pattern resulting from a planar wavefront. Now for every spot position at each grid position *n,m*, we find the difference between the reference and the image under test along the *x* and *y* direction between them and with equation 3.2 we determine the slope of the wavefront at each position of the grid ($\mu m/mm$):

$$S_{n.m}^{X} = \frac{\bar{x}_{n,m}(ref) - \bar{x}_{n,m}(test)}{fml} \tag{3.7}$$

Later in Chapter 4, we will see that this process is not unique for determining the slopes of the wavefront.

3.2.2 Zonal Reconstruction

The term *zonal reconstruction* comes from the fact that we determine the wavefront only "*locally over a limited zone*" [26]. This zone in our case represents the rectangular grid corresponding to the lenslets array of the Shack-Hartmann sensor [25]. The first step for any zonal reconstruction method is to determine what slope model follow our slopes measurements. The main 3 geometries for the slope model correspond to Hudgin, Fried and Southwell geometry. They are depicted in the following figure:

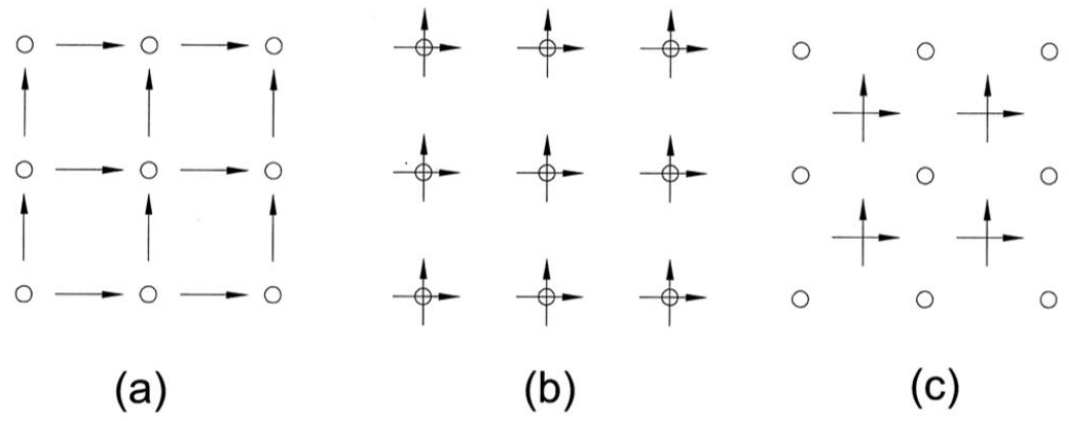


Fig-21: *Wavefront estimation geometries. (a) Hudgin geometry, (b) Southwell geometry, (c) Fried geometry. The white dots correspond to the wavefront phase values and the arrows to the slope measurements. [30]*

These geometries lead to a specific relationship between the slopes and the wavefront phase. The application for each model depends on the grid pattern used, which, in turn, depends on where the actual slope measurements are taken [9]. For instance, Southwell geometry is ideal for a Shack-Hartmann sensor, where each lens measures both the *x* and *y* slopes at the same points.

For the Hudgin geometry, assuming a same number of phase points in the *x* and *y* direction we have N^2 phase points and $2N(N-1)$ slope measurements. The equations relating the slopes and the phase points are:

$$S_{n.m}^{X} = \frac{W_{n+1,m} - W_{n,m}}{D} \; ; \; n = 1, N-1 \; and \; m = 1, N \tag{3.8}$$

$$S_{n.m}^{Y} = \frac{W_{n,m+1} - W_{n,m}}{D} \; ; \; n = 1, N \; and \; m = 1, N-1 \tag{3.9}$$

Where D is the diameter of the lens. Similarly, we can derive the equation for the Fried geometry:

$$S_{n.m}^{X} = \frac{(W_{n+1,m} + W_{n+1,m+1})\backslash 2 - (W_{n,m} + W_{n,m+1})\backslash 2}{D} \tag{3.10}$$

$$S_{n.m}^{Y} = \frac{(W_{n,m+1} + W_{n+1,m+1})\backslash 2 - (W_{n,m} + W_{n+1,m})\backslash 2}{D} \tag{3.11}$$

Finally, for Southwell geometry the relation is the following:

$$\frac{S_{n+1.m}^{X} + S_{n.m}^{X}}{2} = \frac{W_{n+1,m} - W_{n,m}}{D} \; ; \; n = 1, N-1 \; and \; m = 1, N \tag{3.12}$$

$$\frac{S_{n.m+1}^{Y} + S_{n.m}^{Y}}{2} = \frac{W_{n,m+1} - W_{n,m}}{D} \; ; \; n = 1, N \; and \; m = 1, N-1 \tag{3.13}$$

Once we have described the main three geometries for slope measurements, we will discuss different methods for zonal wavefront reconstruction.

LINEAR INTEGRATION

This is the most basic zonal wavefront reconstruction method. We begin at one edge of the wavefront slope data and define the phase value at each integration area as zero. The height of the wavefront in the next adjacent integration area is calculated with the previous phase value, following the Hudgin geometry discussed above. Mathematically, this is given in the x direction by [27]:

$$W_{n,m}^{x} = W_{n-1,m}^{x} + S_{n-1.m}^{X} D \tag{3.14}$$

where D is again the diameter of the lens. This is also performed in the y direction using an identical formula. After this linear integration is performed along both axes, the total wavefront is calculated by:

$$W = W^{x} + W^{y} \tag{3.15}$$

This method could be a good starting point because it is very simple and fast. However, the results are often very noisy, so it is the recommended approach.

LEAST-SQUARE SOLUTION

Having a set of measurements for the slopes S^x and S^y and a model for them (equations 3.8 to 3.13), it is straightforward to apply least-square techniques [9],[28]. In case of equation 3.8 to 3.11 we can write them with matrix notation as:

$$\mathrm{S} = A\,W \tag{3.16}$$

Where S is a column vector containing all the slope measurements in the x and y direction (x firsts), W corresponds to the phase values of the wavefront of length N^2 in a column vector shape, and A is a rectangular matrix of size N^2 x number of slope measurements, that represents the slope model. The standard least square solution is:

$$\mathrm{W} = [A^T A]^{-1}\,[A^T]\,S \tag{3.17}$$

As an example, take a wavefront of size 4x4 that we want to reconstruct. Assuming Hudgin geometry, we have a set of 12 slope measurements in the x direction i.e. $N(N-1)$, and a set another 12 for the y direction. Such scenario is depicted in figure 22.

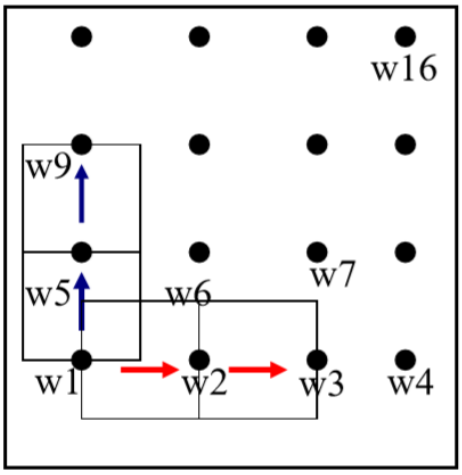


Fig-22: *Example scenario of a wavefront 4x4 for least-square solution. [29]*

In that scenario, the resulting equation 3.16 with the corresponding A matrix that represents the geometry in this case is the following:

$$\begin{bmatrix} s_1^x \\ s_2^x \\ s_3^x \\ s_4^x \\ s_5^x \\ s_6^x \\ s_7^x \\ s_8^x \\ s_9^x \\ s_{10}^x \\ s_{11}^x \\ s_{12}^x \\ s_1^y \\ s_2^y \\ s_3^y \\ s_4^y \\ s_5^y \\ s_6^y \\ s_7^y \\ s_8^y \\ s_9^y \\ s_{10}^y \\ s_{11}^y \\ s_{12}^y \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} x \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{15} \\ w_{16} \end{bmatrix}$$

Fig-23: *Matrix problem on the scenario depicted at figure 22, for the Hudgin geometry. The matrix S in this case is size 24x1, matrix W is size 1x16 and matrix A is size 24x16.*

However, the standard least-square solution is not applicable because $[A^T A]$ is singular what means it can't be inverted.

Our desired solution is the one that has minimum norm [9], which has zero mean. By adding an extra row to S with value 0 and an extra row to A with ones, we assure that our solution has zero mean. Moreover, A becomes not singular, so equation 3.17 can be applied to solve for W.

For Fried geometry, matrix A has different structure but essentially is the same procedure. However, for Southwell geometry the formulation is slightly different:

$$\mathrm{HW} = CS \tag{3.18}$$

Where H is the same A matrix for Hudgin geometry and C can be obtained by changing the sign of the coefficient -1 in A and multiplying by ½. In this case, the least-square solution would be:

$$\mathrm{W} = [H^T H]^{-1} [H^T] CS \tag{3.19}$$

There has been done a lot of research on error propagation for least-square solutions [9], [30]. Indeed, the vector of slope measurements S is noisy, and can be written as:

$$S = S_0 + N \tag{3.20}$$

Where S_0 is the vector for the true slope values and N is the measurement noise. If we call ϵ to the error induced in the wavefront reconstruction by N, then we can write the wavefront vector as:

$$\mathrm{W} = W_0 + \epsilon \tag{3.21}$$

Where W_0 is the true wavefront vector. To quantify the error, we define the error propagation coefficient η as the ratio of the mean variance of the wavefront estimation error σ_w^2 to the variance of the slope measurement error σ_n^2,

$$\eta = \frac{\sigma_w^2}{\sigma_n^2} \tag{3.22}$$

The mean variance of the wavefront estimation error σ_w^2 is given by:

$$\sigma_w^2 = \|\epsilon\|_2^2/m \tag{3.23}$$

Where $\|\epsilon\|_2^2$ is the Euclidean norm and m is the total number of grid points (m = t x t for a square grid and t is the grid size). Given that, it has been shown that the error dependence on the grid size t is logarithmic [30]. The following table includes the expression of the error coefficient for the main three geometries:

Geometry	Error coefficient
Hudgin	$\eta = 0.561 + 0.103\ln(t)$
Fried	$\eta = 0.6558 + 0.3206\ln(t)$
Southwell	$\eta = -0.10447 + 0.2963\ln(t)$

Table 3: *Error propagation coefficient for the main three slope models or wavefront geometries. [30]*

Figure 24 shows how Southwell geometry exhibits lower error propagation than the other geometries when t small, but the Hudgin geometry tends to be a slightly superior when t > 30.

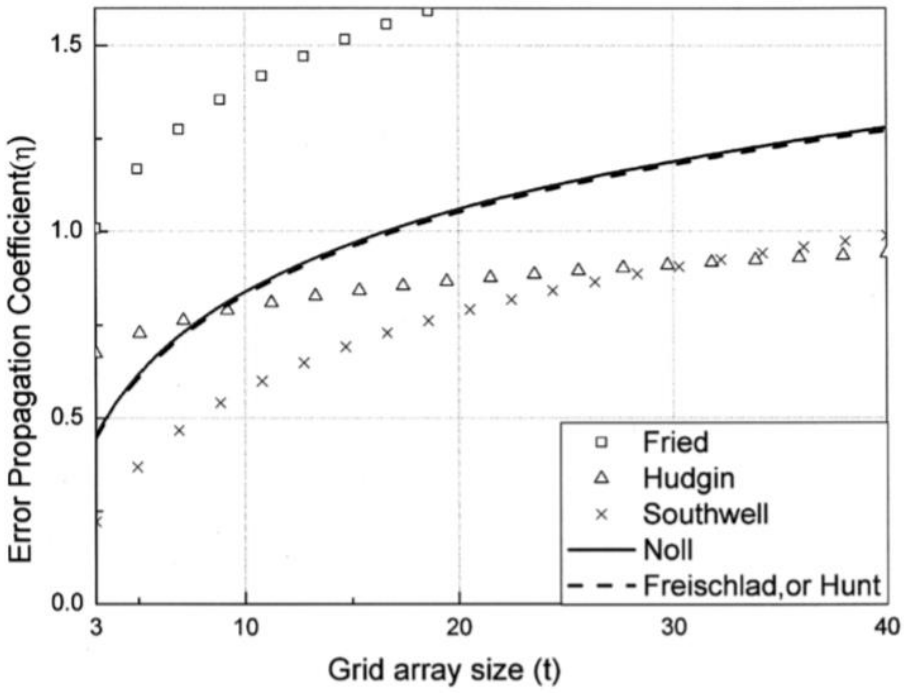


Fig- 24: *Comparation of the error propagation coefficient for different geometries when applying least-square solution. It is depicted that the Southwell geometry is superior to the rest of geometries until t>30. [30]*

3.2.3 Modal Reconstruction

In modal reconstruction, the wavefront surface is described in terms of a "*set of smoothly varying modes*" [25]. This mode fitting can be expressed by:

$$W(x,y) = \sum_{k=0}^{M} c_k n_k F_k(x,y) \tag{3.24}$$

where c_k are the coefficients to be estimated, n_k are normalization factors and $F_k(x,y)$ is the set of orthogonal functions used for fitting our slope measurements. By differentiating equation 3.24 we end up with:

$$S^x = \sum_{k=0}^{M} c_k n_k \, \partial F_k(x,y) / \partial x \tag{3.25}$$

$$S^y = \sum_{k=0}^{M} c_k n_k \, \partial F_k(x,y) / \partial y \tag{3.26}$$

Various types of orthogonal functions can be used to describe our wavefront. Research has been made on fitting wavefront data to Fourier polynomials, Chebyshev polynomials or splines [31], [32]. However, the most stablished basis functions are the Zernike polynomials discussed in previous sections. We saw the Zernike polynomials describe physical properties of the wavefront, specifically optical aberrations. As the main objective of this study is to characterize optical aberrations, we will describe modal method for Zernike polynomials.

There are different methods for modal reconstruction, but we will only focus on difference Zernike polynomial fitting [33]. A summary for modal reconstruction methods is depicted in the following figure:

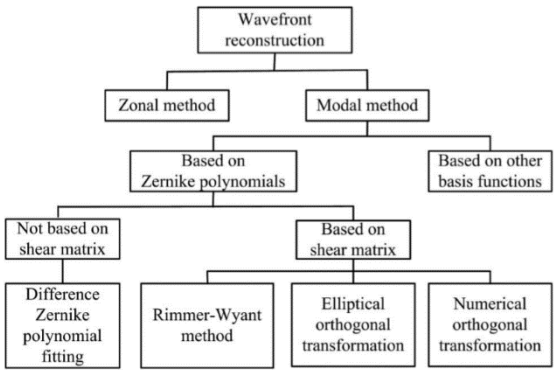


Fig-25: *Different methods for wavefront reconstruction. [33]*

Equations 3.25 and 3.26 can be written in discrete matrix form as (ignoring normalization factors):

$$S^x = D_{zx}C \tag{3.27}$$

$$S^y = D_{zy}\,C \tag{3.28}$$

where S^x and S^y are our set of slope measurements and D_{zx} and D_{zy} are matrices containing the partial derivatives for the Zernike polynomials for the *x* and *y* direction. These are matrices of M columns (number of Zernike modes to fit) and $2N^2$ rows in case of assuming Southwell geometry. Each of the columns contains the partial derivatives of each Zernike polynomial considered. By combining both equations, we get:

$$S = D_z C \tag{3.29}$$

Now standard least-square solution can be applied and obtain the coefficients C:

$$\mathrm{C} = [D_z{}^T D_z]^{-1} D_z{}^T C \tag{3.29}$$

It is important to recall that Zernike polynomials are orthogonal only over the unit circle, so in order to apply modal reconstruction using this set of functions our slope measurements must be defined over a circle. This is not a problem since our data is adjusted to a circular pupil. Moreover, we can derive that the first coefficient (piston) won't be determine by this way. But we do not need to be concerned about the piston coefficient. All the other terms in the expansion have zero mean, and so will have the wavefront. In this way, we satisfy a minimum norm solution.

In terms of noise propagation and error it has been shown that modal reconstruction is superior to zonal [9],[30],[32],[25]. In the following figure a graph depicts how the noise coefficient evolves with grid size in modal reconstruction compared with zonal.

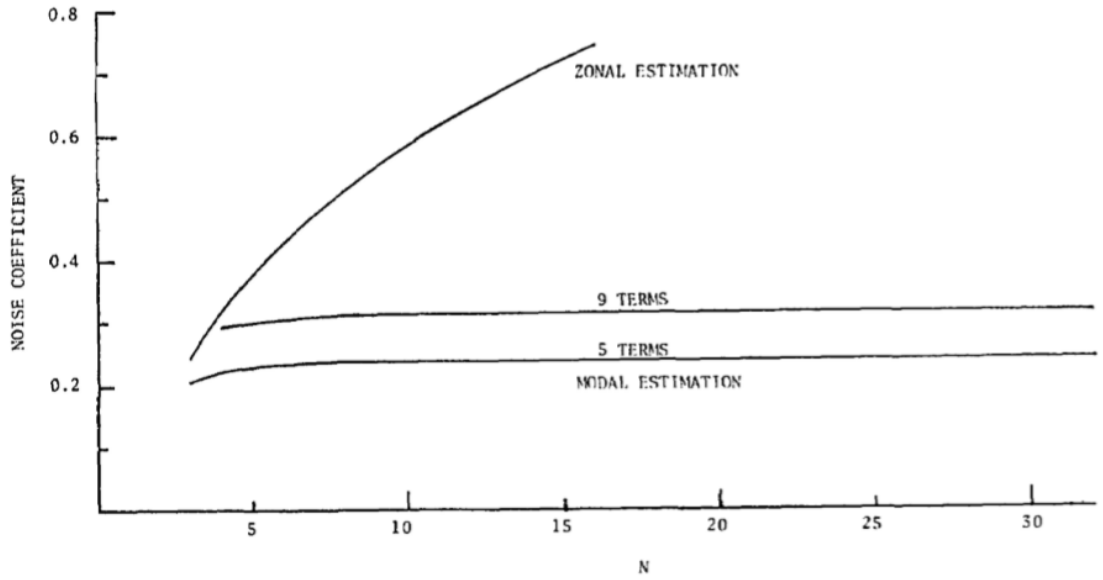


Fig-26: *Graph showing the noise coefficient discussed previously* $\boldsymbol{\eta}$ *for modal and zonal estimation. The results shown are obtained from Southwell geometry. The x axis corresponds to the size of the grid. [9]*

3.3 Zernike Decomposition

Wavefront characterization ends when the surface is set to a specific number of Zernike polynomials, that will represent the aberrations present in the system. Wavefront fitting to Zernike polynomials is only applicable for zonal methods, because in modal methods the result is directly the Zernike coefficients, which is the objective of our wavefront sensing system. Recalling equation 2.13, we can write it in discrete matrix form as:

$$W = ZC \tag{3.30}$$

where Z is a matrix for which each column corresponds to the sampled Zernike functions and W and C are column vectors containing the samples (that we have obtained with our zonal estimation) and coefficients of the wavefront respectively. Again, W must contain only values of the wavefront fitting a circle because Zernike polynomials are not orthogonal outside the unit circle. Notice that in order to compute the coefficients, a higher number of samples N^2 are required than the number of coefficients M to estimate. Least-squares fit to data is then applied:

$$C = (Z^T Z)^{-1} Z^T W \tag{3.31}$$

Chapter 4

Improving Wavefront Sensing

In this chapter, different approaches for improving the functionalities of wavefront sensing and reconstruction are examined. These techniques will be implemented in the MATLAB software, although in this section we restrict to only present the theoretical principles of these techniques. The actual software implementation is discussed in chapter 5.

4.1 Reducing noise

Minimizing error is a top priority in any system. In case of the SH sensor, we saw in section 3.2.2 that the error in the reconstruction of the wavefront comes from error propagation due to noisy slope measurements (equation 3.20). At the same time, these noisy slope measurements come from the centroiding process to find the position of the light spots within the image captured by the sensor. Classic centroiding (equation 3.6) is optimum when there is no noise affecting the image, or for very high SNR levels. However, this is not commonly the case.

Noise sources of a CCD or CMOS based screen can be summarized as: photon noise (from signal and background light) and read out noise [34]. Read out noise follows a Gaussian distribution whereas photon noise obeys Poisson distribution. However, if the number of photons is greater than 10 on each pixel, then photon noise can be approximated to Gaussian distribution [34]. In astronomy, the photons can be very limited and therefore photon noise can't be approximated to Gaussian, but in case of ophthalmology we assure high light conditions in our measurement. We can write the measuring error variance as:

$$\sigma_{error}^2 = \sigma_{photon}^2 + \sigma_{ron}^2 \tag{4.1}$$

where σ_{photon}^2 is the photon noise and σ_{ron}^2 is the read-out noise. In the case of having only photon noise and modeling light spots as Gaussian, then the error can be written as [35]:

$$\sigma_{error}^2 = \frac{\sigma}{\sqrt{N_{ph}}} \tag{4.2}$$

where N_{ph} is the number of photons. This error represents the lowest achievable centroiding error. It has been shown that for no readout noise, simple centroiding achieves this lower boundary [35].

In this section we will present some well-stablished centroiding algorithms for noise reduction. In particular: Thresholding, Windowing, WCoG, Intensity WCoG, Iterative WCoG and Correlation-based centroiding. These techniques will be implemented in our MATLAB program and the user will be able to adopt the desired technique.

4.1.1 Thresholding

Thresholding is the most basic noise reduction technique that can be implemented. It consists on eliminating pixels in the image that present a level lower than a certain threshold, and therefore are considered as noise [35],[36]. After thresholding, the classic CoG is computed (equation 3.6).

The common way to implement thresholding is by relative to maximum value. We first look for the maximum value on the image I_{max} and the threshold th is set as $T\ x\ I_{max}$, where T can be selected to optimize in each case. For relatively high SNR levels, T is optimum at 0.2 [35]. Then, every intensity value lower than threshold is set to 0:

$$I(x,y) > th \rightarrow \ \ I(x,y) = 0 \tag{4.3}$$

4.1.2 Windowing

Windowing is a variation of classic centroiding. In this method, we define a window of certain length that can be circular, rectangular or other shape [35]. This window is commonly centered at the pixel with maximum intensity on each grid sub area, so that pixels laying outside the window are set to zero [35]. In the following figure an example is depicted for a circular window:

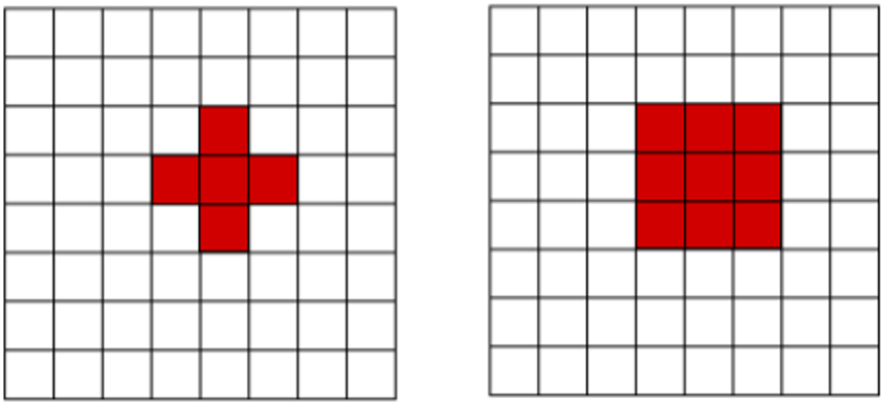


Fig-27: *Circular window representation. On the left, the window has a radius of 1 pixel, whereas on the right of 1.5 pixels. [35]*

4.1.3 WCoG

In weighted center of gravity (WCoG) technique, a weighting function is applied over each sub-area of the grid $W_e(x, y)$ [36]. This function is optimum when it has the shape of the light spot, which is typically Gaussian:

$$W_e(x, y) = A\, x \exp\left[-\frac{(x-x_0)^2}{2\sigma_x^2} - \frac{(y-y_0)^2}{2\sigma_y^2}\right] \tag{4.4}$$

where A is the amplitude. The center coordinates of the weighting function x_0 and y_0 can be set at the maximum intensity pixel. The centroid equation becomes:

$$\bar{x} = \frac{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} W_e(i,j) I(i,j) i}{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} W_e(i,j) I(i,j)} s \tag{4.5}$$

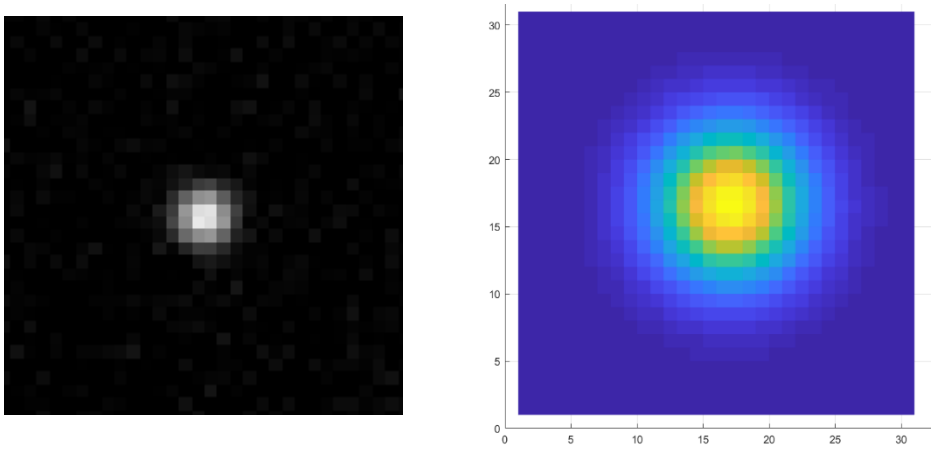


Fig-28: *In the left, example of a captured light spot with noise. In the right, gaussian weighting function generated to apply.*

4.1.4 Intensity WCoG

This method is a variation of the WCoG technique, where the weighting function coincided with the intensity distribution of the spot pattern $W_e(i, j) = I(i, j)$. The method performs a better job when the light levels are low and for low readout and background noise [36]. Equation 3.36 then becomes:

$$\bar{x} = \frac{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} I(i,j)^2 i}{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} I(i,j)^2} s \tag{4.6}$$

We can also modify the weight of the intensity function, in order to optimize the technique for different SNR levels. This will be studied later.

4.1.5 Iterative WCoG

This technique takes WCoG and compute it iteratively in order to compensate inaccurate centroiding and refine the result. After each iteration, the center of $W_e(i,j)$ is changed to the new location after previous computation [36]. The centroid location for the iteration number n (x-direction) is defined as:

$$\bar{x}^n = \frac{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} W_e(i,j)^n I(i,j) i}{\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} W_e(i,j)^n I(i,j)} s \tag{4.7}$$

where,

$$W_e(x,y)^n = A\, x \exp\left[-\frac{\left(x - {x_0}^{n-1}\right)^2}{2\sigma_x^2} - \frac{\left(y - {y_0}^{n-1}\right)^2}{2\sigma_y^2} \right] \tag{4.8}$$

The first iteration the gaussian function is defined by equation 3.35. After each iteration we can also modify the width, but this has little effect on accuracy [36]. This process implies the problems of any iterative process like saturation, convergence and speed.

4.1.6 Correlation-based centroiding

All the methods discussed so far are modifications of the center of gravity algorithm. In this sense, correlation centroiding is radically different. CoG algorithms are quite susceptible to noise and spot abnormalities [37]. Correlation technique has been demonstrated as much less sensitive to noise [37].

The idea behind this method is to find the position of the light spot by calculating the cross-correlation function between a model light spot (template) and the sub-image

corresponding to each grid sub-area. The formula for the discrete normalized cross-correlation function in 2D is:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \overline{f_{u,v}}]\,[t(x-u,y-v) - \bar{t}]}{\{\sum_{x,y}[f(x,y) - \overline{f_{u,v}}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\}^{0.5}} \tag{4.9}$$

where f is the image and t is the template. The cross-correlation function can be also computed in the Fourier domain to optimize efficiency.

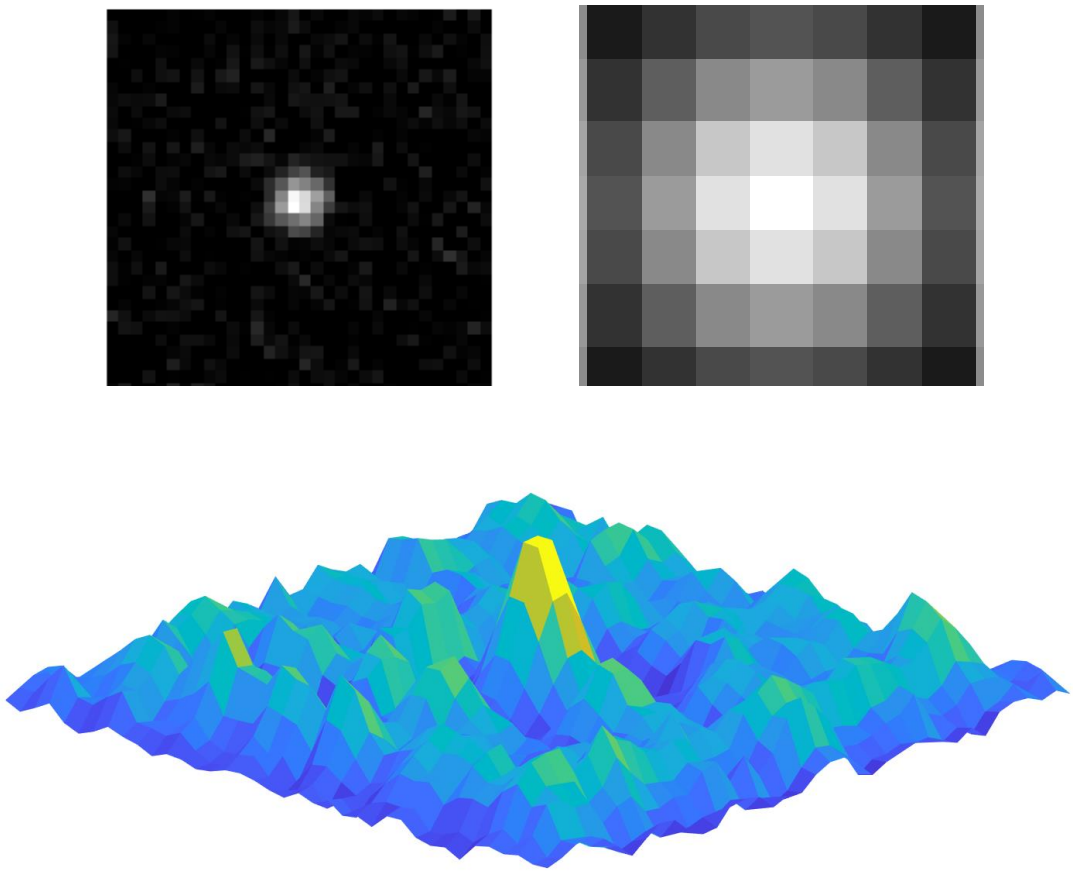


Fig- 29: *Example of a cross correlation calculation. Top left, a noisy sub-image containing a light spot. Top right, gaussian template. Bottom, resulting normalized cross-correlation function.*

After the computation of the cross-correlation function, we can find the maximum value and therefore the best-matching-pixel (BMP). However, pixel accuracy is not enough, so we need to apply interpolation techniques in order to reach sub-pixel accuracy [38]. Taking the neighbors of the BMP, we can apply parabolic interpolation, gaussian, or other methods. In case of gaussian interpolation with 5 neighbors points, we can obtain the sub-pixel displacement from the BMP with [38]:

$$\Delta_x = \frac{ln\, S_{-1,0} - ln\, S_{1,0}}{2[ln\, S_{1,0} - 2ln\, S_{0,0} + ln\, S_{-1,0}]} \tag{4.10}$$

$$\Delta_y = \frac{ln\, S_{0,-1} - ln\, S_{0,1}}{2[ln\, S_{0,1} - 2ln\, S_{0,0} + ln\, S_{0,-1}]} \tag{4.11}$$

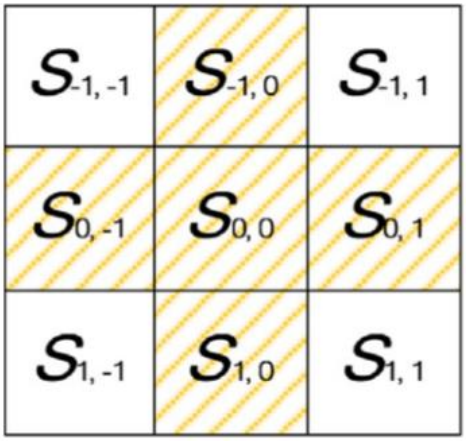


Fig-30: *Scheme of a 5-point interpolation approach. S0,0 corresponds to the BMP. [38]*

4.2 Increasing dynamic range

As we discussed in section 3.1, the Shack-Hartmann sensor consists of a micro-lens array, where each lens gathers lights projecting one spot to the detection screen. Therefore, each lens defines an area on the screen, where the spot is supposed to be situated. In a conventional algorithm, the screen is divided into static sub-areas corresponding to each lens, and then the techniques for centroiding are applied to find the position of the spots. From this, is derived that the maximum spot deviation from the center of the lens area, i.e. the dynamic range of the sensor, is equal to the radius of the lens.

Putting this range into perspective, [39] remarks that this range is "*sufficient for the majority of normal human eyes when measuring aberrations close to the optical axis*". However, the authors of the article also point out that this can fail for such measurements in which "*refractive error is moderate to high, or when pathology is present that distorts the refractive components of the eye*". In addition, the system developed in this work, is not limited to one specific application, so we must consider situations that require higher dynamic range.

The first approach to increase the dynamic range of the Shack-Hartmann sensor is to modify the hardware of the sensor. However, there is an intrinsic trade-off between increasing dynamic range and decreasing sensitivity [40]. Representing the dynamic range as the maximum angle α that the locally plane distorted wavefront can present at each lens,

$$\alpha_{max} = \frac{D}{2f} \tag{4.10}$$

where D is the diameter of the lens and f is the focal length. From this equation we see that in order to increase dynamic range we can either set larger lens diameter or shorter focal length. We saw in section 3.3, that we need a minimum amount of slope measurements in order to determine a concrete number of Zernike modes (redundancy sampling). In this sense, this commonly fixes the diameter of the lens, because it determines the number of lenses, i.e. number of slope measurements, that can be on a given pupil diameter.

That said, reducing the focal length causes inevitably a decrease on sensitivity. Representing the sensitivity as the minimum angle α, that the locally plane distorted wavefront can present at each lens:

$$\alpha_{min} = \frac{\Delta s_{min}}{f} \tag{4.11}$$

where Δs_{min} is the minimum distance detectable in the spot, which is determined *"by the pixel size of the photodetector, the accuracy of the centroid algorithm and the signal-to-noise ratio of the sensor"*. Combing equations 4.11 and 4.10 we can represent the trade-off between sensitivity and dynamic range as:

$$\alpha_{min} = \frac{2\Delta s_{min}\alpha_{max}}{D} \tag{4.11}$$

4.2.1 Sorting Algorithms

As stated at 1.2, we are working with the sensor hardware of Thorlabs, therefore we focus only on software development. In this sense, as an alternative to hardware modification, there exists various algorithm presented by authors that increase the dynamic range without modifying the sensor's hardware.

If we don't restrict the standpoint of the problem to a static grid of sub-areas, we can see it as a sorting problem. Basically, by different image processing tools we can detect the spots of a reference wavefront and a distorted wavefront independently, and then, it is a matter of associating correctly each spot detected to the corresponding lens.

One of the early efforts on solving this problem by sorting algorithms is presented in [41]. Here, authors applied unwrapping algorithms already used in interferograms. Many authors have proposed other solutions for the problem. We find inter alia, B-spline fit, and extrapolation, spiral ordering or Zernike fit and extrapolation [39].

The approach is similar for each of these algorithms. They work under the assumption that aberrations exhibit low spot displacement at the center of the pupil. In this way, starting at the center, the spots can be associated to their corresponding lenses with a conventional algorithm, and then use different techniques for extending these associations to the rest of the pupil through multiple iterations:

Conventional algorithm

1	10	19	36	45	54	63	72	81
2	11	20	37	46	55	64	73	82
3	12	21	38	47	56	65	74	83
4	13	22	39	48	57	66	75	84
5	14	23	40	49	58	67	76	85
6	15	24	41	50	59	68	77	86
7	16	25	42	51	60	69	78	87
8	17	26	43	52	61	70	79	88
9	18	27	44	53	62	71	80	89

Lundström and Unsbo (2004) - "B-spline"

9	8	7	6	5	6	7	8	9
8	6	5	4	3	4	5	6	8
7	5	3	2	1	2	3	5	7
6	4	2	0	0	0	2	4	6
5	3	1	0	0	0	1	3	5
6	4	2	0	0	0	2	4	6
7	5	3	2	1	2	3	5	7
8	6	5	4	3	4	5	6	8
9	8	7	6	5	6	7	8	9

Leroux and Dainty (2009) - "Zernike"

3	3	3	3	3	3	3	3	3
3	2	2	2	2	2	2	2	3
3	2	1	1	1	1	1	2	3
3	2	1	0	0	0	1	2	3
3	2	1	0	0	0	1	2	3
3	2	1	0	0	0	1	2	3
3	2	1	1	1	1	1	2	3
3	2	2	2	2	2	2	2	3
3	3	3	3	3	3	3	3	3

Smith and Greivenkamp (2008) - "Spiral"

78	47	48	49	50	51	52	53	54
77	46	23	24	25	26	27	28	55
76	45	22	7	8	9	10	29	56
75	44	21	6	0	0	11	30	57
74	43	20	5	0	1	12	31	58
73	42	19	4	3	2	13	32	59
72	41	18	17	16	15	14	33	60
71	40	39	38	37	36	35	34	61
70	69	68	67	66	65	64	63	62

Fig-31: *"Order in which lenslets of a square lenslet array are unwrapped for each of the algorithms explored here. The zero entries indicate the initial location of central spots before the main part of the algorithm takes effect. Subsequent numbers indicate iterations of each algorithm". [39]*

4.2.2 Zernike-based sorting

In the software implemented in this thesis, we have chosen to work with the Zernike-based sorting algorithm proposed by Leroux and Dainty [42]. The algorithm is not very complex to implement and uses principles of Zernike polynomials and wavefront reconstruction that are already used in other areas of the system.

The principle of this algorithm is the following. As discussed previousy, prior to apply the algorithm, we need to have identified the spot positions on the screen. Therefore, we start with two sets of spots locations, for the reference spots X_{ref} and the distorted wavefront spots X_{dist}. Typically, the first spots assumed to be correctly laying into the sub-areas of the lenses, correspond to a grid of 3x3. From this grid, we can then get 9 associations between reference and displaced spots, and therefore obtain our first set of slope measurements.

From this set of slope measurements, the process discussed in 3.2.3 for modal reconstruction is applied, obtaining the Zernike coefficients C_0 for that set of slope measurements. In this first, iteration, the reconstruction is used commonly considering only tilts, defocus and astigmatism [42]. The coefficients C_0 represent an estimation of the real coefficients obtained when considering all the area of the pupil. We can extrapolate them and obtain an estimation of the positions of the displaced spots:

$$\tilde{X}_{dist} = X_{ref} + A\ x\ C_0 \tag{4.11}$$

where $\tilde{X}_{dist}$, are the estimated spot locations for the distorted wavefront, and A is a matrix in which each component *"is the shift of a given Shack-Hartmann spot (row index), induced by a given Zernike aberration (column index)"[42].* Once calculated the estimated positions, we search for the nearest actual spot of the sensor and then associate it to the corresponding reference spot. However, the first approximation of C_0 is not accurate, so we can't limit the algorithm to just one iteration.

The first iteration is used to extrapolate the spot locations for the lenses areas around the 3x3 grid. Then, we calculate again the coefficients as done before but with this new slope measurements and the process is repeated until the whole pupil is covered. Authors in [42] propose the following scheme of extrapolation:

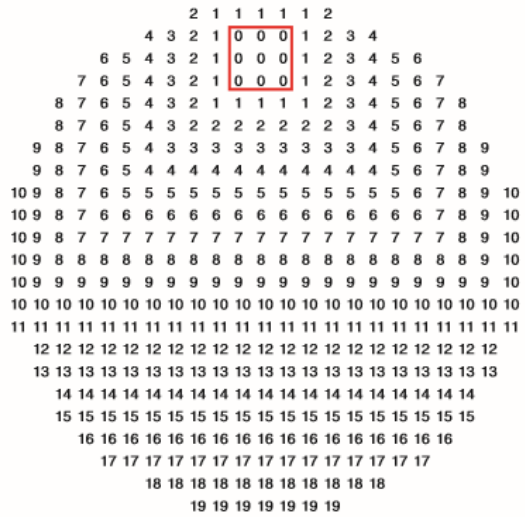


Fig-32: *"Steps for the extrapolation of the centroid positions across the measured pupil". [42]*

Chapter 5

Software Implementation with MATLAB

This chapter is dedicated to discussing the software implementation of the wavefront reconstruction and characterization system, using the background and algorithms presented in chapters 3 and 4. In addition, this chapter tries to be a guide for readers interested on using the software presented. First, the basic structure of the system is explained and consecutively, the implementation of each part is discussed.

5.1 System Structure and Parameters

The wavefront reconstruction system is implemented in MathWorks MATLAB. This platform offers a multiparadigm programming environment, and it is especially convenient for scientific and engineering projects, due to its multiple built-in toolboxes and active ecosystem. It is worth mentioning that in this case, strict structured programming has been used. Having said that, the different parts in which the system can be divided are: Parameters definition, Pupil Definition and Spot Sorting, Centroid calculation and Slope Measurements, Wavefront Reconstruction and Statistics and Results.

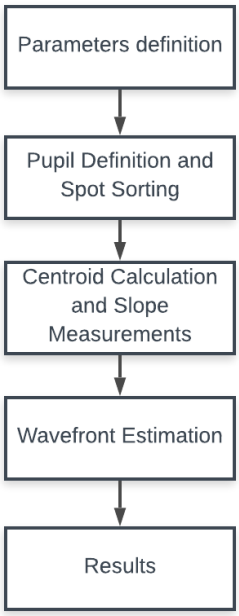


Fig-33: *Diagram of the structure of the system.*

The folders and functions' structure of the MATLAB system is the following:

- **Root Directory**: Here we find the launching script Run.m, the principal functions and the sub-folders. The main functions are:
    - calculateCentroid
    - calculateLocalGradients
    - calculateZernikeCoeffs
    - calculateZernikeFunctions
    - definePupil
    - modalReconstructWF
    - showResults
    - zonalReconstructWF
    - sortSpots
- **additionalFunctions**: This folder contains secondary functions. Specifically:
    - drawArrow
    - fitLocalGradients
    - getSouthwellMatrix
    - RMSE
    - shiftFix
    - solveMatrixSystem
- **externalFunctions**: In this directory there are the functions that are taken from the file exchange at Mathworks.com. Therefore, code of this functions is not developed in this thesis and is used under authors copyright:
    - customgauss
    - inpaint_nans
    - mplot
    - intgrad2
- **simulationToolBox**: Here the simulation tools developed for testing techniques of wavefront reconstruction are stored:
    - getDistortedWavefront
    - testNoiseReductionTechnique
    - testRangeExtensionTechnique
- **wavefrontFiles**: This folder stores the input files of any input data coming from the device used for sensing the wavefront.

The script **Run.m** is the starting point of the system. The parameters are defined and then the corresponding functions are called to execute each step as depicted in figure 33. These parameters model the hardware of the sensor and define multiple characteristics of the estimation of the wavefront. They are stored in a "struct" data type and will be passed to any function that requires them.

Run.m

```
%% Clear variables,close windows and set environment
dbstop if error;
clearvars;
close all;
addpath('./externalfunctions');
addpath('./additionalFunctions');
addpath('./wavefrontFiles');
addpath('./simulationToolBox');
get(0,'Factory');
set(0,'defaultfigurecolor',[1 1 1]);

%% Set parameters and load images
parameters = struct();
parameters.centroid_technique = 5;
parameters.ignore_sorting = false;
parameters.sort_method = 1;
parameters.zonal_method = 4;
parameters.spot_px_sep = 32;
parameters.radius_definition =
'auto';%'user_text_defined','user_click_defined','auto'
parameters.zernikeorder = 5;
parameters.pixel_spacing = 4.65;
parameters.focal_length = 5.2;
parameters.lens_diameter = 0.150;
parameters.wavelength = 0.543;
parameters.input = 'd45_raw.bmp';
parameters.calibration = 'calibration.bmp';
parameters.winSize = 3;
parameters.inWeight = 2;
parameters.WCoG_var = 4;
parameters.IWCoG_it = 10;
parameters.size_template = 9;
parameters.var_template = 2;

ref_wf = imread(parameters.calibration);ref_wf =
im2double(ref_wf(:,:,1));
input_wf = imread(parameters.input);input_wf =
im2double(input_wf(:,:,1));
 %% Pupil definition and Spot Sorting
[input_wf_centered,ref_wf_centered,parameters] =
definePupil(input_wf,ref_wf,parameters);
[c_rdata_sorted,c_idata_sorted] =
sortSpots(input_wf_centered,ref_wf_centered,parameters);

%% Centroid calculation and Slope Measurements
localGradients =
calculateLocalGradients(input_wf_centered,ref_wf_centered,c_rdata_sor
ted,c_idata_sorted,parameters);

%% Wavefront estimation
%Zonal Reconstruction
```

```
[X,Y,WF] = zonalReconstructWF(localGradients,parameters);
%Zernike Decomposition
[Z,ZernikeFunctions,unitcircle] =
calculateZernikeFunctions(parameters.zernikeorder,parameters.gridSize
);
ZernikeCoeffs = calculateZernikeCoeffs(Z,WF(unitcircle));
%Modal Reconstruction
[ZernikeCoeffsModal,WFmodal] =
modalReconstructWF(ZernikeFunctions,localGradients,unitcircle,paramet
ers);

%% Results
showresults(input_wf_centered,X,Y,WF,ZernikeFunctions,ZernikeCoeffs,u
nitcircle);
showresults(input_wf_centered,X,Y,WFmodal,ZernikeFunctions,ZernikeCoe
ffsModal,unitcircle);
```

Listing 1: *Run.m*

Parameter	**Description**
centroid_technique	Noise reduction technique used
ignore_sorting	Boolean to ignore the sorting algorithms and use a conventional algorithm
sort_method	Sorting method used for associating reference spots and displaced spots
zonal_method	Method used for zonal reconstruction
spot_px_sep	Separation between 2 reference spots in pixels
radius_definiton	Method for defining the radius of the pupil
zernikeorder	Order of the Zernike functions used for the wavefront characterization
pixel_spacing	Size of a pixel in μm
focal_length	Focal length of the lenses in mm
lens_diameter	Diameter of the lenses in mm
wavelength	Wavelength of the beam
input	Filename of the image of the distorted wavefront to reconstruct
calibration	Filename of the referent wavefront used for calibration
winsize	Size in pixels for the window used in windowing noise cancelling
inweight	Power of the weighted intensity noise cancelling technique
wcog_var	Variance of the gaussian function used for weighted center of gravity technique
iwcog_it	Number of iterations used in iterative WCoG
size_template	Size of the template used in correlation-based centroiding
var_template	Variance of the template used in correlation-based centroiding

Table 4: *List of parameters of the system.*

5.2 Pupil Definition and Spot Sorting

In this section, the pupil is defined selecting only the portion of the image corresponding to the domain of reconstruction. Then, the spots are sorted using the method set in the parameters.

PUPIL DEFINITION

The image captured by the sensor represents the full micro lens array. However, the area corresponding to the pupil domain of our measurement is only a portion of that image. Therefore, we need to define a pupil in the full image and crop that specific area. In order to define the pupil, different strategies can be used. This is represented by the parameter ***radius_definition*** and can take the following values:

- **'auto':** In this case, the spots are detected by the function 'regionprops', provided by the MATLAB Image Processing Toolbox. By this function, the tentative locations of the spots in the image of the distorted wavefront are obtained. Then the maximum distance along the x and y axis is determined. This distance represents the diameter of the pupil. The center of the pupil is estimated by averaging the detected spots.
- **'user_click_defined':** With this option the user defines the center of the pupil and radius by clicking at the wished locations in the image.
- **'user_text_defined':** In this case, the center is defined by averaging spot locations as in 'auto' and radius is set by text input by user. The radius is defined by the number of spots wished to be in the pupil domain and therefore is dependent on the parameter *spot_px_sep*.

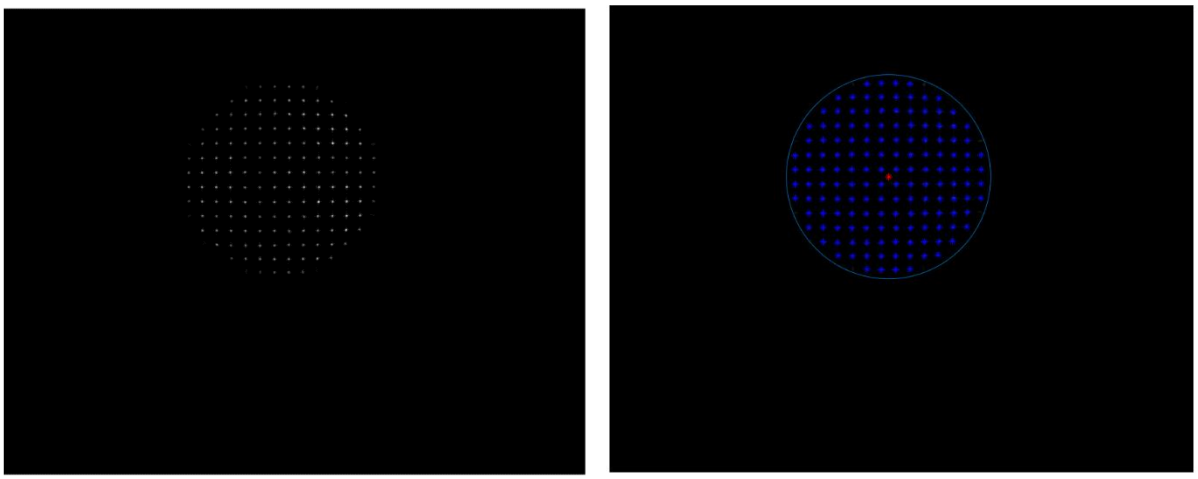


Fig-34: *Example of pupil definition in 'auto'. In the right, raw image captured by the sensor. In the left, tentative spot locations(blue) and computed center (red) by averaging.*

It is worth mentioning that the radius is set as a number of light spots to consider from the center of the pupil. This in turn determines the size of the grid. For instance, if a radius of 7 spots from center has been calculated or selected, then the size of the square grid corresponding to the working domain will be 14x14. At each position of the grid, a reference and displaced spot will be found, and a local slope will be measured. Although the size of the grid is square, only positions belonging to the pupil (circular shape) will be considered for the wavefront reconstruction.

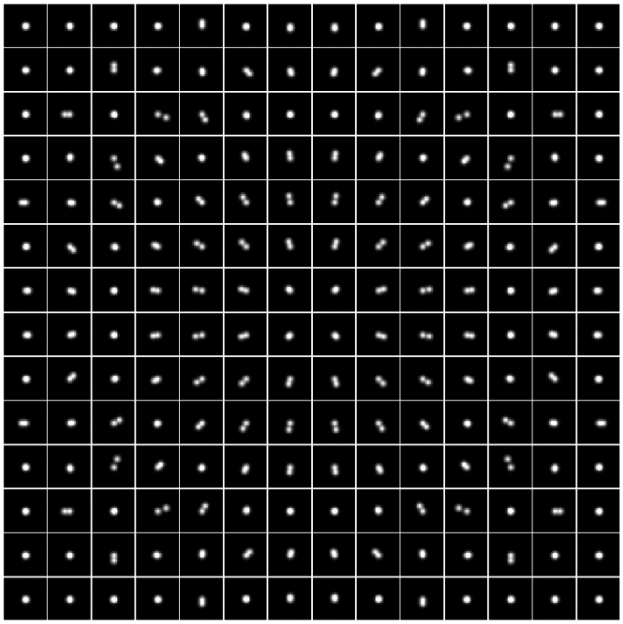


Fig-35: *Example of a grid of size 14x14. In this example, a spherical aberration can be seen.*

Once the pupil has been defined (center and radius), the next step is to crop the portion of the image corresponding only to the pupil domain, for both the reference image and the distorted. However, both images may not be aligned, resulting in wrong wavefront estimation. To fix this, after cropping the images, they are shifted iteratively in order to reduce the misalignment. It is remarkable that by doing this, the tilt and tip coefficients determined are not representative, and therefore the information about them is lost.

Nonetheless, this is not a problem because these coefficients are not relevant for us, as they don't represent an optical aberration. That said, the functions involved in the pupil definition are ***definePupil*** as main function and ***shiftFix*** as additional function. Table 5 includes the list of inputs and outputs of the function definePupil and a description of them.

Input	Description
input_wf	Raw image captured by the sensor corresponding to the distorted wavefront to characterize
ref_wf	Reference image from de sensor used for calibration
parameters	Struct of the parameters of the sensor
Ouput	**Description**
input_wf_centered	Cropped image of the distorted wavefront corresponding to the pupil domain defined in the function
ref_wf_centered	Cropped image of the reference wavefront corresponding to the pupil domain defined in the function
parameters	Struct of the parameters of the sensor, with added fields (spots_from_center, pupil_radius and gridSize).

Table 5: *Inputs and outputs of the function definePupil.*

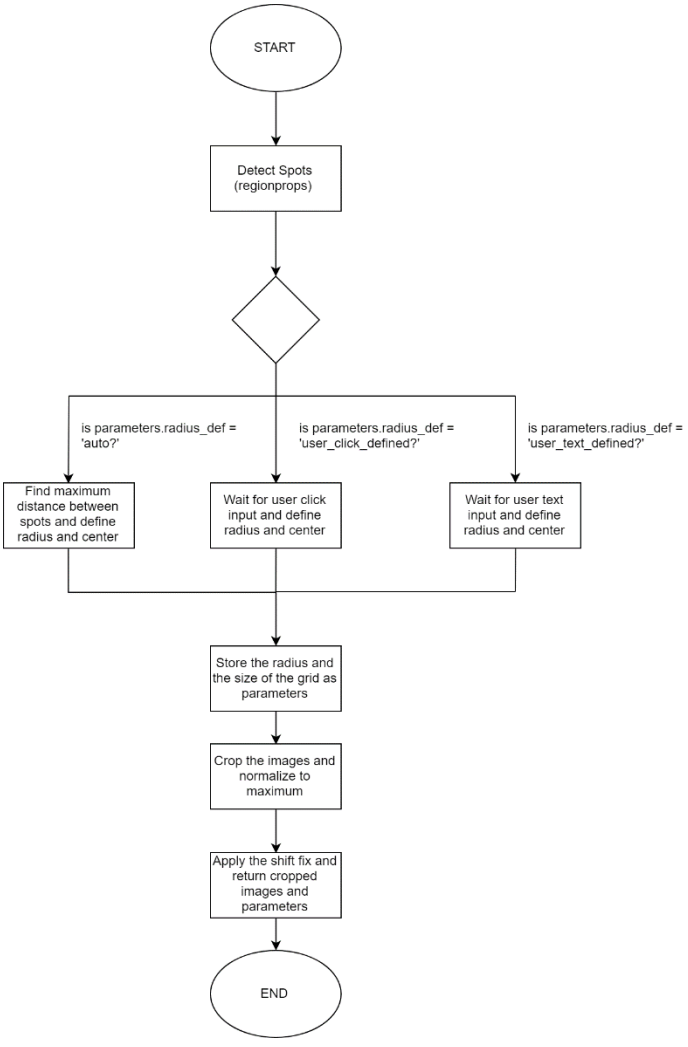


Fig-36: *Flux diagram of the pupil definition* ***(definePupil)****.*

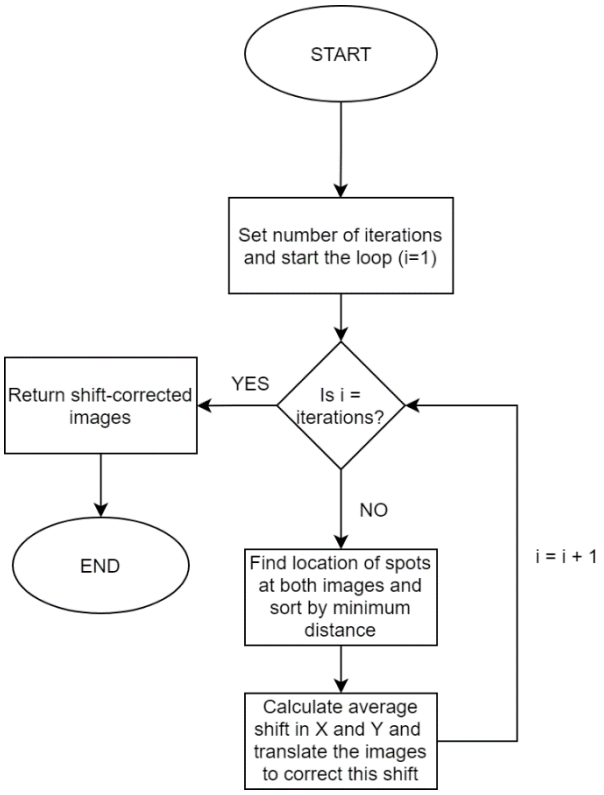


Fig-37: *Flux diagram of the algorithm for correcting misalignment between reference image and distorted image (**shiftFix**).*

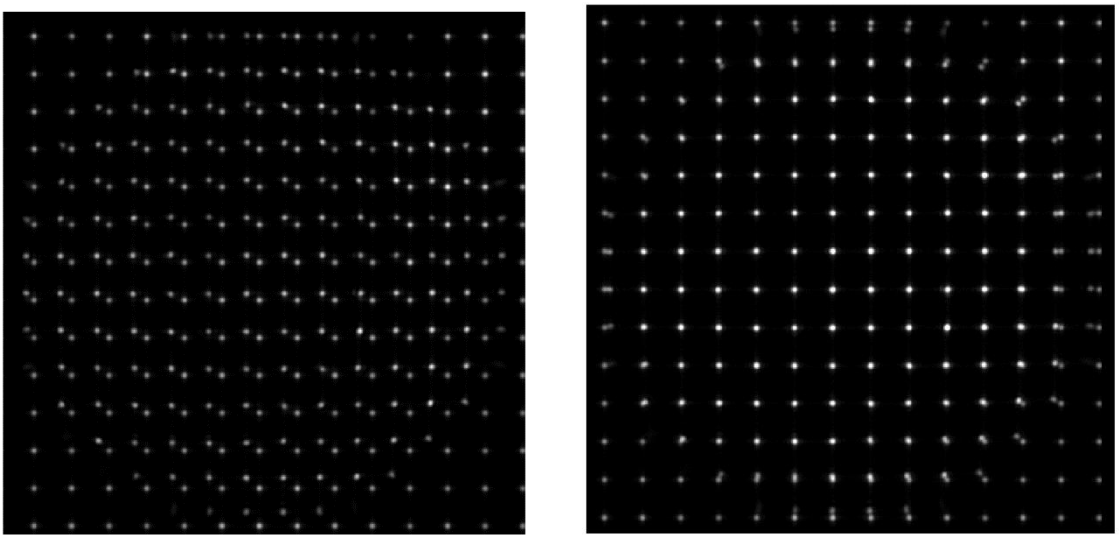


Fig-38: *Example of misaligned images. On the right, cropped reference and distorted image superimposed. On the right, same images but with the alignment fixed.*

SORT SPOTS

After defining the pupil, cropped and aligned images are returned. These images contain the spots relevant for the wavefront estimation. The next step consists on associating each spot between images in order to find the local slopes. Notice that after sorting the spots, centroiding techniques must be applied. The locations of the spots obtained here are only tentative. The function executed to sort the spots is ***sortSpots***, and it implements two methods, selected by the parameter ***sort_method***.

- **Minimum Distance**: The basic sorting method by default uses minimum Euclidean distance. This method increases slightly the dynamic range as will be shown in the results. The algorithm is depicted in the following flowchart:

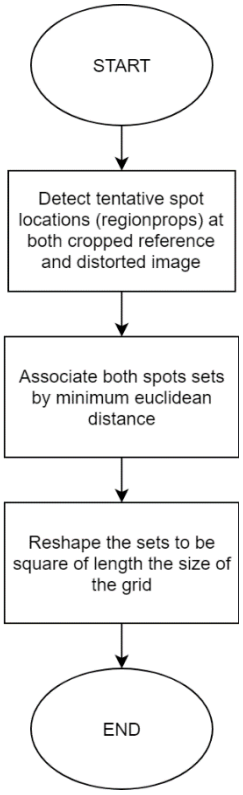


Fig-39: *Flowchart of sorting spots by minimum distance.*

- **Zernike-based sorting:** This corresponds to the sorting algorithm presented at 4.2.2. In this implementation, the spots are first sorted by minimum distance, i.e. previous method. Then, sorted spots situated in the middle of the pupil are selected for the first iteration. The first set of spots is a grid of size 2x2. Now Zernike coefficients are calculated by modal reconstruction algorithm from the slopes measured in this grid 2x2. From these coefficients we estimate the position of the surrounding spots to the first grid 2x2, associate the estimated positions with the closest real ones and repeat the process iteratively.

This algorithm will extend significantly the dynamic range of the sensor, but we can still extend it more by adding another processing stage. It may happen that two estimated spot positions are associated to the same real spot because distance was minimum for both. Therefore, one of them is not associated correctly. However, after finishing the previous algorithm and cover all the grid, most of the spots are correctly associated.

To fix this, those spot associations that are performed more than once are eliminated and the coefficients are estimated again without these associations. Then, we try to associate again spots that were eliminated and check if all the associations are one to one. If not, we repeat this process until is fixed, or until the maximum number of iterations is achieved, what will mean that the distortion is too high for the algorithm. In figure 41, a flowchart depicts the algorithm in detail.

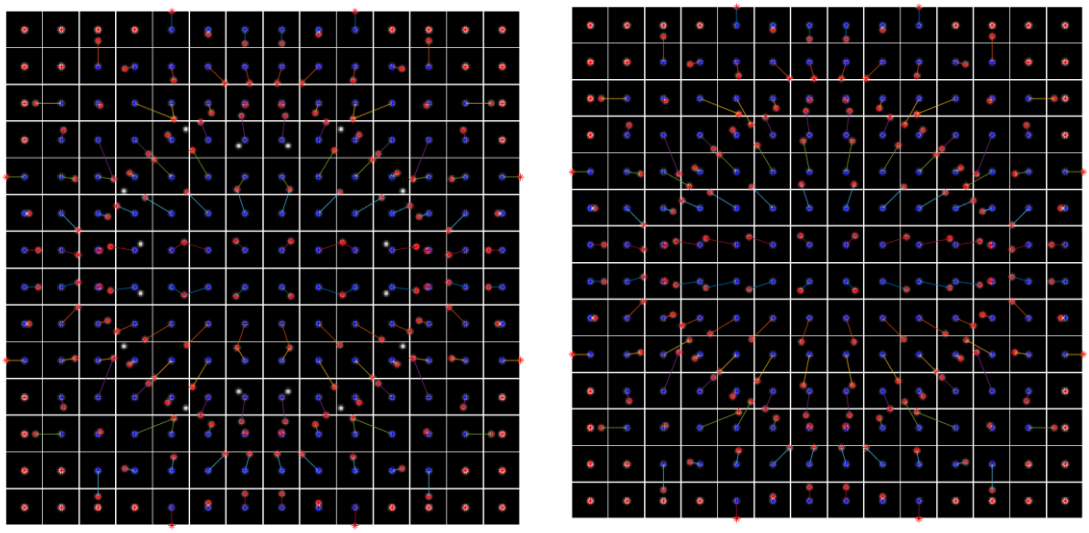


Fig-40: *Example of the two stages of the Zernike sorting algorithm. Image on the left depicts a high distorted wavefront where the basic algorithm has failed to associate correctly reference and displaced spots. Some reference spots (blue) have been associated with the same displaced spot (red). On the right, the scond stage of the algorithm has been conducted. Now every spot is associated correctly to its counterspot.*

Input	**Description**
Ref_wf_centered	Cropped image of the reference wavefront corresponding to the pupil domain defined in the function
Input_wf_centered	Cropped image of the distorted wavefront corresponding to the pupil domain defined in the function
Ouput	**Description**
C_rdata_sorted	Cell array of size gridSize x gridSize, containing the spot locations of the reference image sorted
C_idata_sorted	Cell array of size gridSize x gridSize, containing the spots locations of the distorted image sorted

Table 6*: Inputs and outputs of sortSpots.*

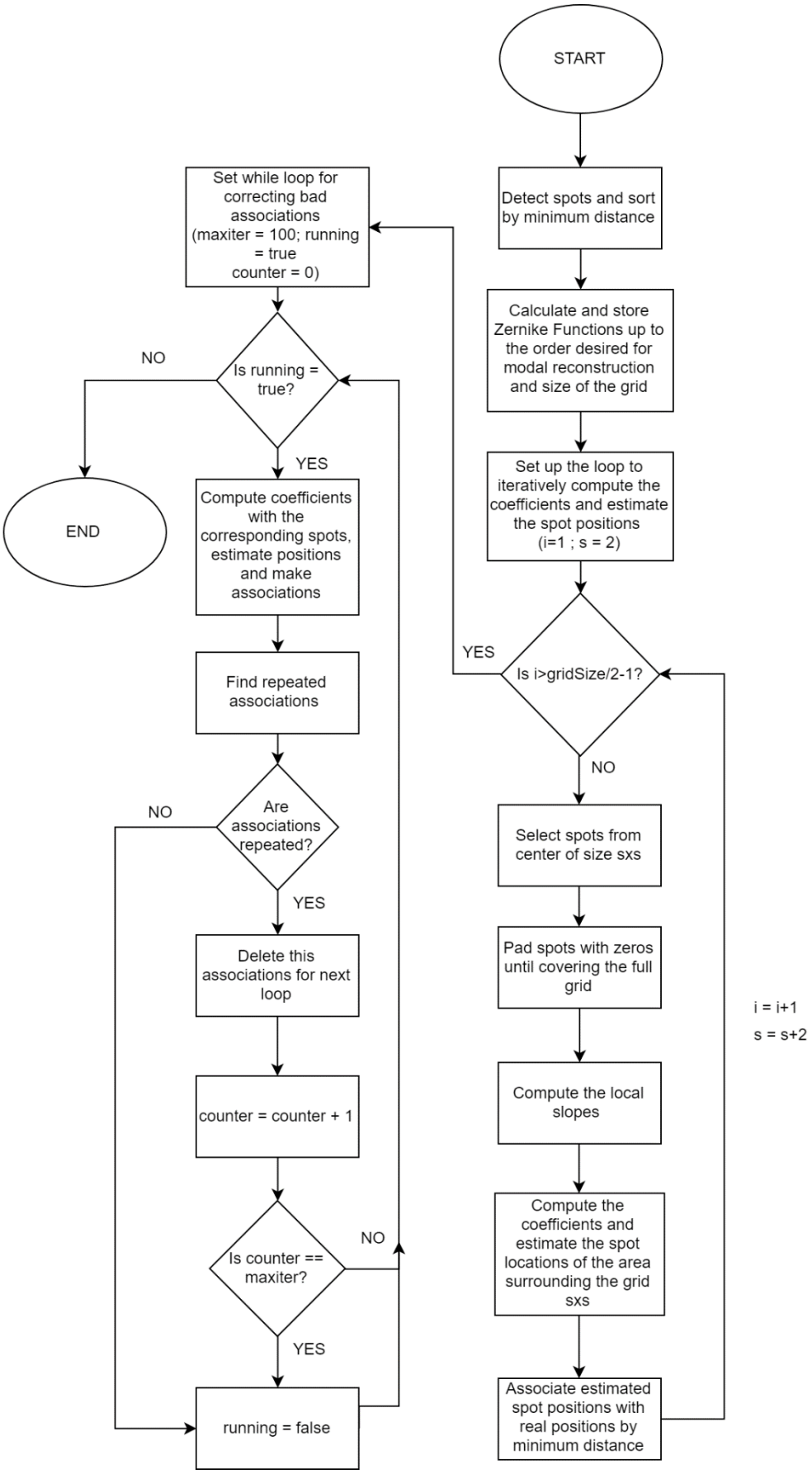


Fig-41: *Flowchart of the Zernike sorting algorithm.*

5.3 Centroid Calculation and Slope Measurements

Although the spots are sorted, as mentioned previously, the positions stored are tentative and won't be used for measuring the slopes of the wavefront. The "real" position of the spots will be determined by using the techniques described in Chapter 4, which are selected in the parameters (centroid_technique). There is also the option to ignore the previously done sorting and use a conventional algorithm.

Once the real positions are calculated, by simple subtraction the slopes are found at each grid position. These slopes will only have a value within the grid positions that contain spots for the distorted wavefront, i.e. pupil domain. The rest of the grid will be extrapolated to get more accurate results in case of zonal estimation. This is because in zonal estimation all the slopes in the square grid play a role in the final wavefront estimation. However, for modal estimation, only the pupil domain is affecting the solution. This extrapolation is done with the function external function **inpaint_nans**, which uses methods based on sparse linear algebra and PDE discretizations. The main function of this section is **calculateLocalGradients**.

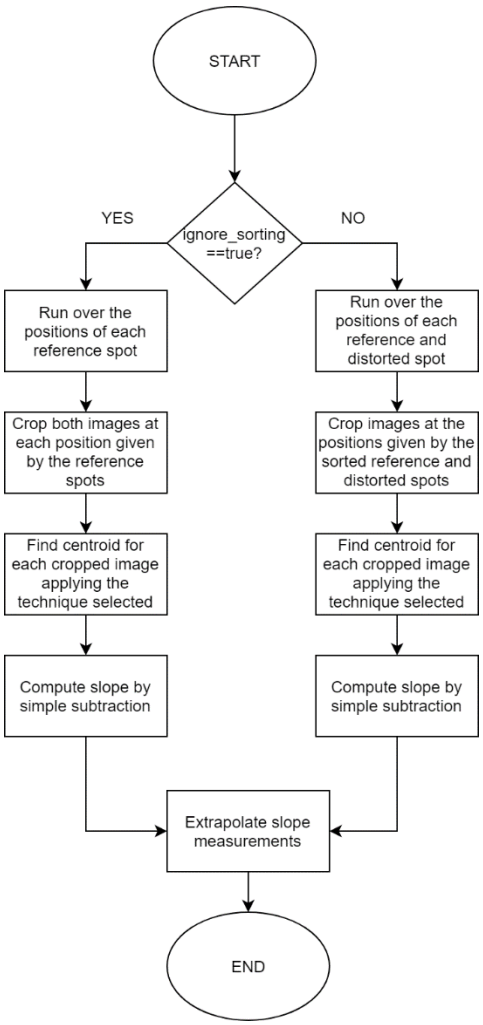


Fig- 42: *Flowchart of the function calculateLocalGradients.*

Input	Description
ref_wf_centered	Cropped image of the reference wavefront corresponding to the pupil domain defined in the function
input_wf_centered	Cropped image of the distorted wavefront corresponding to the pupil domain defined in the function
c_rdata_sorted	Cell array of size gridSize x gridSize, containing the spot locations of the reference image sorted
c_idata_sorted	Cell array of size gridSize x gridSize, containing the spots locations of the distorted image sorted
parameters	Struct of the parameters of the sensor
Ouput	**Description**
localGradients	Struct containing the value of the local slopes for the X and Y directions

Table 7: *Inputs and outputs of calculateLocalGradients.*

The function **calculateCentroid** is responsible of computing the centroid of each cropped image corresponding to each grid sub-area (micro-lens area), following the technique specified on the parameter ***centroid_technique*** for noise cancelling. Table 7 shows how to select each technique:

Value of centroid_technique	Technique
1	No noise reduction
2	Thresholding
3	Windowing
4	WCoG by Intensity
5	WCoG by Gaussian
6	Iterative WCoG be Gaussian
7	Correlation
8	Adaptive Template Correlation

Table 8: *Values for the parameter centroid_technique.*

The implementation of the first 7 methods is straightforward and won't be discussed here (see appendices). In case of correlation, is remarkable to discuss our particular implementation, i.e. number 8. Recalling section 4.1.6, the correlation technique is based on computing the cross-correlation function between a spot template and the image where the centroid is to be obtained. It results that, the more similar the template is to the real spot, the more accurate the centroid estimation will be. This template is in our case a gaussian function.

In this sense, the implementation proposed here calibrates the template for each image processed. In this way, the template will match better the spot present in the image and the centroid computed will be more accurate. In this implementation, the parameters calibrated are the size and the variance of the template. However, this technique would be also suitable to model spots that differ from gaussian due to diffraction effects. In Chapter 6, a comparison between the classic correlation method and this implementation will be shown.

In order to calibrate the template, the procedure is simple: The template starts with the values set by default in the parameters at the beginning. Then, calibration for either the size or variance of the template is performed. First, the cross-correlation function is computed between image being processed and template and the maximum value of the cross-correlation is stored. The parameter of the template under calibration is then modified by a pre-defined interval, the cross-correlation is computed again, and the maximum value is stored. Now the maximum value of this iteration is compared with the previous one. The outcome of this comparison will determine keep changing the parameter under calibration until it reaches saturation. This algorithm is depicted in figure 43.

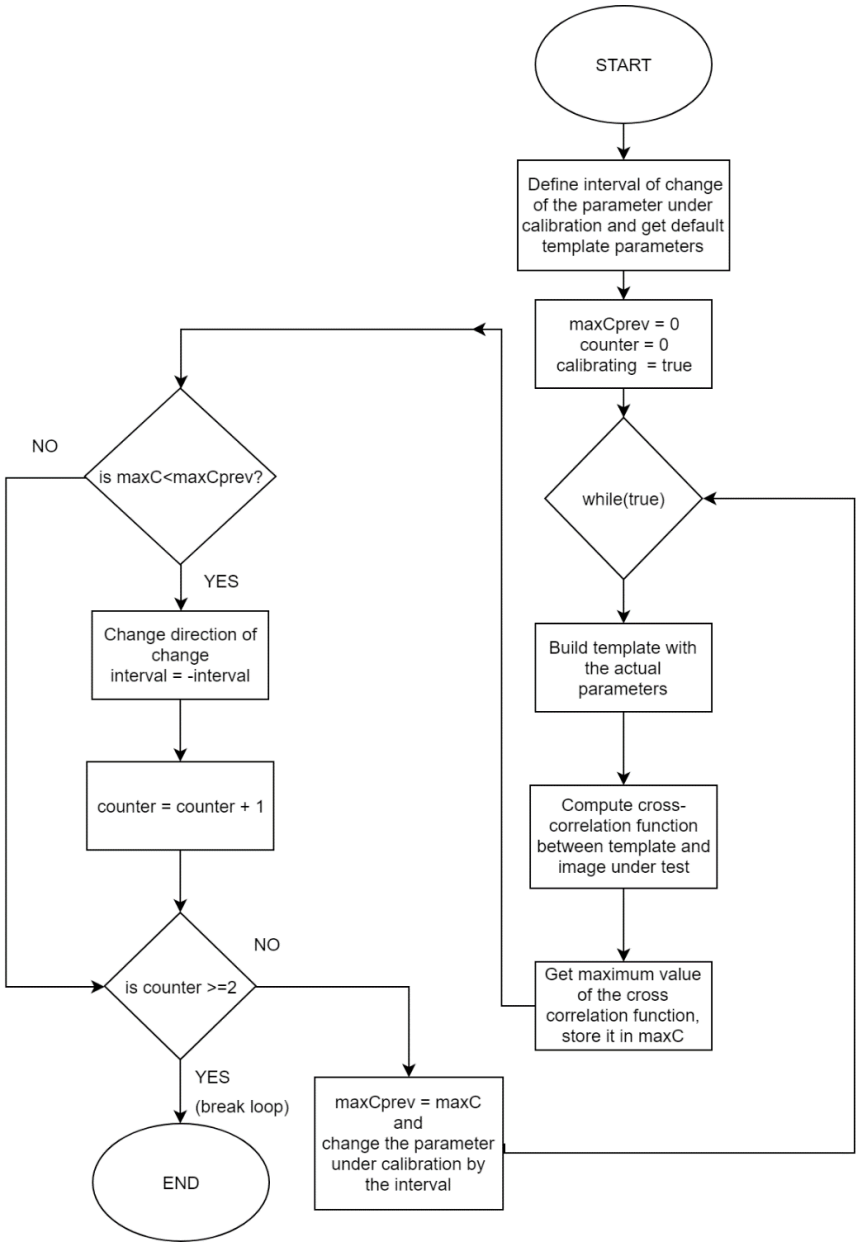


Fig-43: *Flowchart of the algorithm used to calibrate a certain parameter of the template used in correlation-based centroiding.*

5.4 Wavefront Reconstruction

In this section the wavefront is reconstructed and characterized. The output of the previous stage is a struct called localGradients, that contains the slope values obtained at each grid position. From this slope values, is straightforward to apply the reconstruction methods discussed in Chapter 3.

5.4.1 Zonal Reconstruction

For zonal reconstruction approaches, different options are given. The parameter zonal_method controls which one is performed. Table 9 shows the values that zonal_method can take. The recommended option in this case is to use the Least-Square solution using Southwell Geometry. Alternatively, the external function intgrad2 to integrate gradient data of 2 dimensions also gives consistent results. Functions involved in the zonal reconstruction are **zonalReconstructWF**, **getSouthwellMatrix**, **solveMatrixSystem** and **intgrad2**.

Value of zonal_method	**Method**
1	Linear integration starting from the borders (not recommended)
2	Linear integration starting from the center (not recommended)
3	Least Squares solution (Hudgin Geometry) (Not Recommended)
4	Least Squares solution (Southwell Geometry) (Recommended)
5	Use external function intgrad2

***Table 9**: Options for zonal reconstruction.*

Input	**Description**
localGradients	Struct containing the value of the local slopes for the X and Y directions
parameters	Struct of the parameters of the sensor
Ouput	**Description**
X	2D-coordinates of the wavefront for the X axis
Y	2D-coordinates of the wavefront for the Y axis
WF	Values of the reconstructed wavefront over the grid

***Table 10:** Inputs/Outputs of the function zonalReconstructWF.*

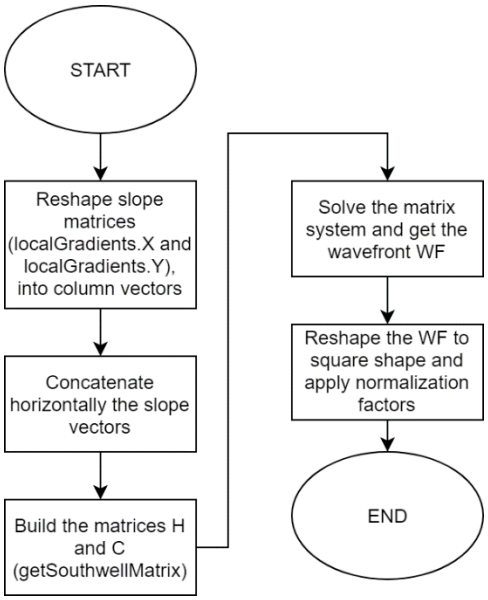


Fig-44: *Algorithm of the Southwell zonal reconstruction method.*

In order to apply equations 3.17, 3.19 and solve the matrix system, the function **solveMatrixSystem** is used. In this case, 3 different options are given and are shown at Table 11.

Option of solveMatrixSystem	Description
1	Standard least-squares solution using mldivide
2	Least-Squares solution using lsqminnorm
3	Least-Squares solution using singular value decomposition SVD (pinv)

Table 11: *Options for the function solveMatrixSystem.*

The first option uses the MATLAB function **mldivide**, which is a general approach for solving matrix systems. It takes advantage of the symmetries of the matrices to compute the inverse more efficiently in terms of computational effort. The solution given by this first option doesn't satisfy minimum norm, and in general is not recommended to use here. Second option uses the MATLAB function **lsqminnorm**, which gives minimum norm solution, and uses complete orthogonal decomposition **(COD)**. This is usually the recommended option.

For those cases where the matrix to invert is singular or the rank is deficient, the approach to follow is the third one, where the MATLAB function **pinv** is used. This computes the Moore-Pseudo Inverse, by singular value decomposition **(SVD)**, assuring minimum norm solution for those matrices that can't be inverted by a normal procedure.

5.4.2 Zernike Decomposition

To decompose the wavefront obtained by zonal reconstruction algorithms into the Zernike functions, first the polynomials must be pre-computed. The parameter *zernikeorder* controls the order the polynomials to consider for decomposition, as input of the function **calculateZernikeFunctions**.

Input	**Description**
order	Order of the Zernike functions
L	Size of the grid to compute the functions
Ouput	**Description**
Z	Matrix with much columns as polynomials computed. Each column contains the values of each polynomial in the unit circle
ZernikeF	Cell array containing the polynomials
unitcircle	Logical matrix of size L indicating which positions belong to the unit circle.

Table 12: *Inputs/Outputs of calculateZernikeFunctions.*

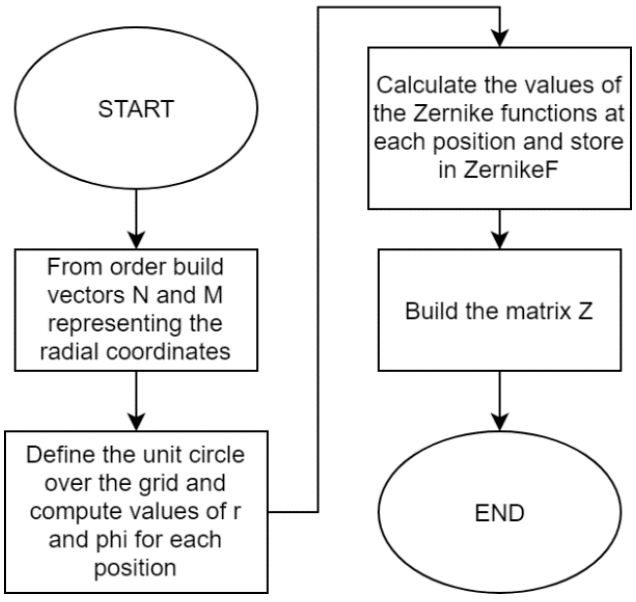


Fig-45: *Flowchart of the function calculateZernikeFunctions.*

Two aspects must be commented here: normalization factor and unit circle. There are multiples normalization factors that can be applied. In this case, the following is used:

$$norm = \sqrt{\left(1 + \delta_{ij}\right) x \ (n+1)/\pi} \tag{5.1}$$

Where δ_{ij} is the Kronecker delta. This factor to make the integral of $\sqrt{r\ x\ Znm(r, phi)^2}$ equal to 1.

Regarding the unit circle, is important to notice how the definition is done. Usually, cartesian coordinates are defined between -1 and 1, and then a conversion to polar coordinates is performed to obtain the unit circle. In this way, positions at the borders won't belong to the unit circle defined, as the radius will be higher than 1. This results in ignoring the values situated at the edges of the wavefront for the Zernike estimation, and also in case of modal reconstruction. If those values are desired to be taken into the decomposition, higher radius must be selected at the definition of the pupil, or definition of the unit circle must be changed.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
2	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
3	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
4	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
5	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
6	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
7	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
8	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
9	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
10	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
11	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
12	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
13	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1
14	-1	-0.8462	-0.6923	-0.5385	-0.3846	-0.2308	-0.0769	0.0769	0.2308	0.3846	0.5385	0.6923	0.8462	1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.4142	1.3100	1.2163	1.1358	1.0714	1.0263	1.0030	1.0030	1.0263	1.0714	1.1358	1.2163	1.3100	1.4142
2	1.3100	1.1966	1.0933	1.0030	0.9295	0.8771	0.8496	0.8496	0.8771	0.9295	1.0030	1.0933	1.1966	1.3100
3	1.2163	1.0933	0.9791	0.8771	0.7920	0.7298	0.6966	0.6966	0.7298	0.7920	0.8771	0.9791	1.0933	1.2163
4	1.1358	1.0030	0.8771	0.7615	0.6617	0.5858	0.5439	0.5439	0.5858	0.6617	0.7615	0.8771	1.0030	1.1358
5	1.0714	0.9295	0.7920	0.6617	0.5439	0.4485	0.3922	0.3922	0.4485	0.5439	0.6617	0.7920	0.9295	1.0714
6	1.0263	0.8771	0.7298	0.5858	0.4485	0.3264	0.2433	0.2433	0.3264	0.4485	0.5858	0.7298	0.8771	1.0263
7	1.0030	0.8496	0.6966	0.5439	0.3922	0.2433	0.1088	0.1088	0.2433	0.3922	0.5439	0.6966	0.8496	1.0030
8	1.0030	0.8496	0.6966	0.5439	0.3922	0.2433	0.1088	0.1088	0.2433	0.3922	0.5439	0.6966	0.8496	1.0030
9	1.0263	0.8771	0.7298	0.5858	0.4485	0.3264	0.2433	0.2433	0.3264	0.4485	0.5858	0.7298	0.8771	1.0263
10	1.0714	0.9295	0.7920	0.6617	0.5439	0.4485	0.3922	0.3922	0.4485	0.5439	0.6617	0.7920	0.9295	1.0714
11	1.1358	1.0030	0.8771	0.7615	0.6617	0.5858	0.5439	0.5439	0.5858	0.6617	0.7615	0.8771	1.0030	1.1358
12	1.2163	1.0933	0.9791	0.8771	0.7920	0.7298	0.6966	0.6966	0.7298	0.7920	0.8771	0.9791	1.0933	1.2163
13	1.3100	1.1966	1.0933	1.0030	0.9295	0.8771	0.8496	0.8496	0.8771	0.9295	1.0030	1.0933	1.1966	1.3100
14	1.4142	1.3100	1.2163	1.1358	1.0714	1.0263	1.0030	1.0030	1.0263	1.0714	1.1358	1.2163	1.3100	1.4142

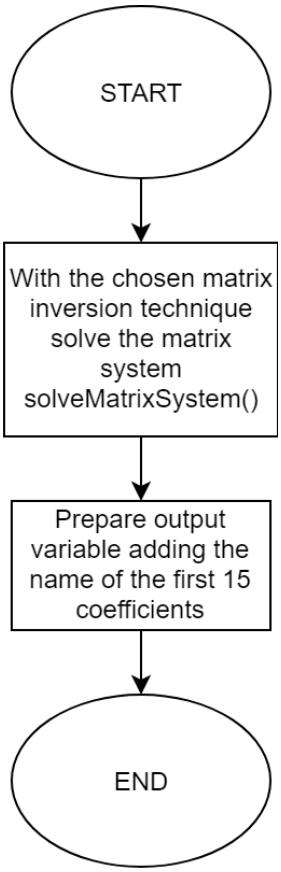
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	1	1	0	0	0	0
3	0	0	1	1	1	1	1	1	1	1	1	1	0	0
4	0	0	1	1	1	1	1	1	1	1	1	1	0	0
5	0	1	1	1	1	1	1	1	1	1	1	1	1	0
6	0	1	1	1	1	1	1	1	1	1	1	1	1	0
7	0	1	1	1	1	1	1	1	1	1	1	1	1	0
8	0	1	1	1	1	1	1	1	1	1	1	1	1	0
9	0	1	1	1	1	1	1	1	1	1	1	1	1	0
10	0	1	1	1	1	1	1	1	1	1	1	1	1	0
11	0	0	1	1	1	1	1	1	1	1	1	1	0	0
12	0	0	1	1	1	1	1	1	1	1	1	1	0	0
13	0	0	0	0	1	1	1	1	1	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figu-46: *Example of a normal unit circle definition over a square grid of size 14x14. At the top, the cartesian coordinates for the x axis over the grid. At the middle, in polar coordinates (radius r). Notice how in the borders the radius is higher than one, and therefore the unit circle logical (bottom), excludes those values.*

Once the Zernike functions are calculated and the unit circle is defined, is straightforward to apply equation 3.31 and obtain the coefficients of the wavefront. The function performing this step is **calculateZernikeCoeffs**.

Input	**Description**
Z	Matrix, for which each column contains the values of a Zernike function over the unit circle
WF	Column vector containing the values of the wavefront estimated with zonal reconstruction algorithm that belong to the unit circle
Ouput	**Description**
ZernikeCoeffs	Computed coefficients

Table 13: *Inputs/Outputs calculateZernikeCoeffs.*



Figu-47: *Flowchart of the function calculateZernikeCoeffs.*

5.4.3 Modal Reconstruction

Modal reconstruction is conducted after the zonal reconstruction and decomposition. It takes advantage of previously used functions, like calculateZernikeCoeffs, and calculateZernikeFunctions. The main function responsible of this section is **modalReconstructWF**.

Input	**Description**
ZernikeFunctions	Zernike functions calculated previously at the decomposition stage
localGradients	Struct containing the slope measurements
unitcircle	Logical matrix indicating grid positions belonging to the unit circle
Ouput	**Description**
ZernikeCoeffsModal	Zernike coefficients obtained from the modal algorithm
WF	Reconstructed WF from the coefficients

Table 14: *Inputs/Outputs of modalReconstructWF.*

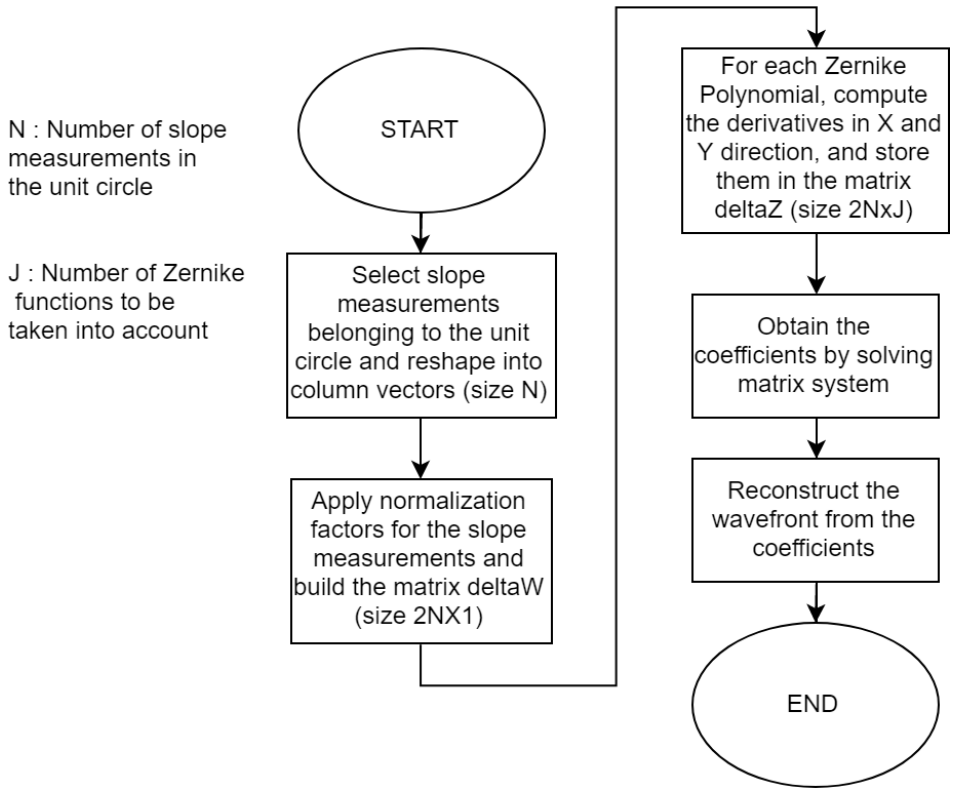


Fig-48: *Flowchart of the function modalReconstructWF.*

5.5 Statistics and Results

After reconstructing the wavefront and obtaining the coefficients, the characterization of the optical aberrations present in the optical system is done. Last stage of the program is dedicated to show the results. The wavefront, coefficients and original image are displayed together. In addition, the calculated Zernike functions are displayed in a multiplot figure. An example is shown in figure 49. Functions involved are showResults and mplot as external function.

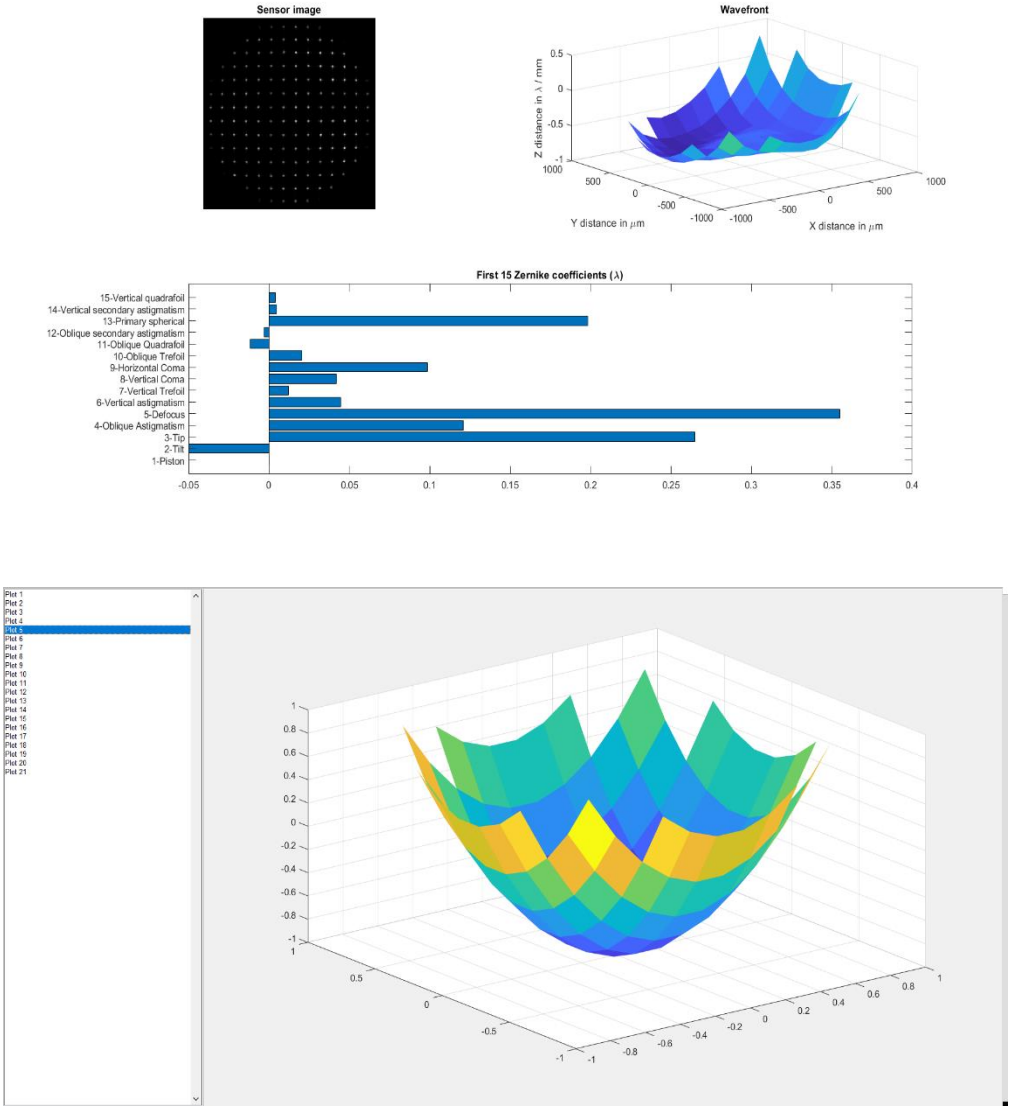


Fig-49: *Example of displaying the results of the system. (showResults). The Zernike coefficients displayed correspond to the first 15. The rest are stored in their corresponding variable. Same display as top example is shown for both zonal and modal reconstruction.*

5.6 Simulation Environment

Recalling Chapter 1, one of the objectives of this thesis is to implement and test the different techniques on a simulated system. For that, a simple simulation toolbox is given with the following functions: **getDistortedWavefront, testNoiseReductionTechnique, testRangeExtensionTechnique**.

The main function of the toolbox is **getDistortedWavefront**. This function is responsible for generating simulated sensor images of distorted wavefronts. To do so, the approach is very simple. The Zernike coefficients of the distorted wavefront to be generated are given as input or generated randomly. Then, the wavefront is reconstructed from the coefficients, matching the size of the desired grid. Once the wavefront is reconstructed, inverse process to the previous discussed system is performed to obtain the positions of the spots on the simulated image for both reference and distorted image. In this sense, the characteristic of any sensor can be modeled by defining the parameters in the same way as we did previously.

Finally, in these positions, simulated gaussian spots are placed. To generate these simulated light spots the external function **customgauss** is used. Size and variance can be set as desired to modify the shape of the simulated light spot. The result of this function corresponds to the images that would come out from a real sensor, if the system to characterize had the aberrations represented by the input coefficients, with no noise.

Input	**Description**
Z	Zernike functions grouped in columns
ZernikeFunctions	Cell array containing the Zernike Functions
unitcircle	Logical matrix indicating grid positions belonging to the unit circle
Parameters	Parameters of the sensor
C	Variable of the coefficients. It can either be a string "random" or a vector
Ouput	**Description**
WF_ref	Simulated wavefront obtained by the coefficients
C_ref	Zernike coefficients
idist	Simulated sensor image corresponding to the distorted wavefront
iref	Simulated reference image

Table 15: *Inputs/Outputs of getDistortedWavefront.*

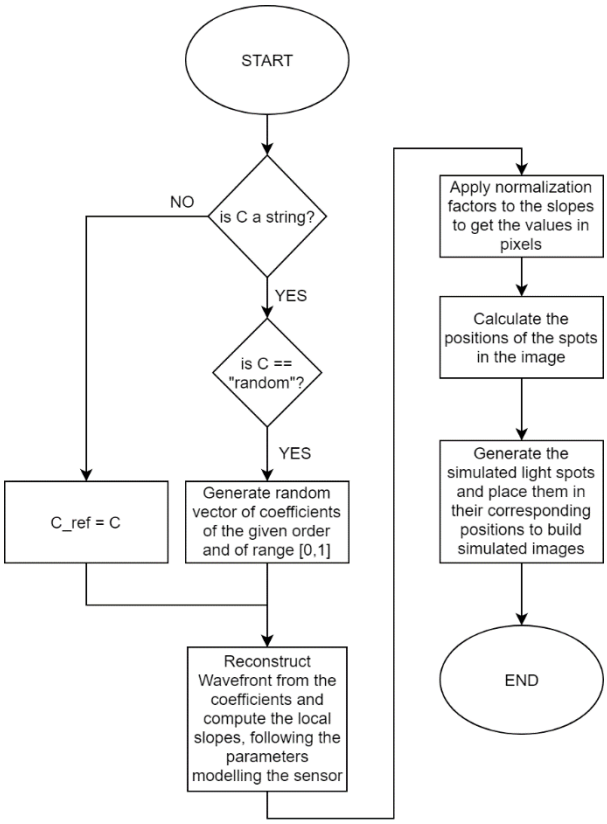


Fig- 51: *Flowchart of getDistortedWavefront.*

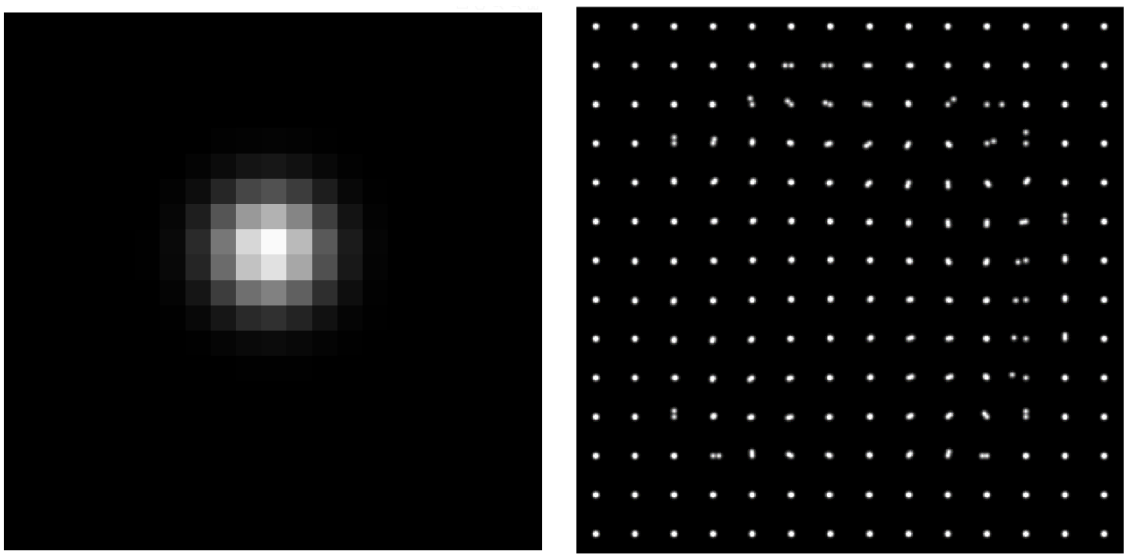


Figure 50: *On the left, simulated gaussian spot. On the right, simulated image of a randomly generated wavefront.*

The functions **testNoiseReductionTechnique** and **testRangeExtensionTechnique** technique uses the aforementioned function to generate images and conduct a test.

In case of testNoiseReductionTechnique, noise is added to the images to fulfill a certain SNR level and then the specified centroiding technique is applied. The wavefront is reconstructed by the modal algorithm and compared with the simulated reference. This comparison is done by computing the RMSE.

Input	**Description**
Parameters	Parameters of the sensor
technique	Centroiding technique to use in the reconstruction
snr	Level of SNR (in other words, noise to add)
meanV	Mean of the noise to add
Ouput	**Description**
RMS_WF	RMSE between the reference wavefront and the reconstructed after adding noise and applying the noise reduction technique specified

Table 16: *Inputs/Outputs of testNoiseReductionTechnique.*

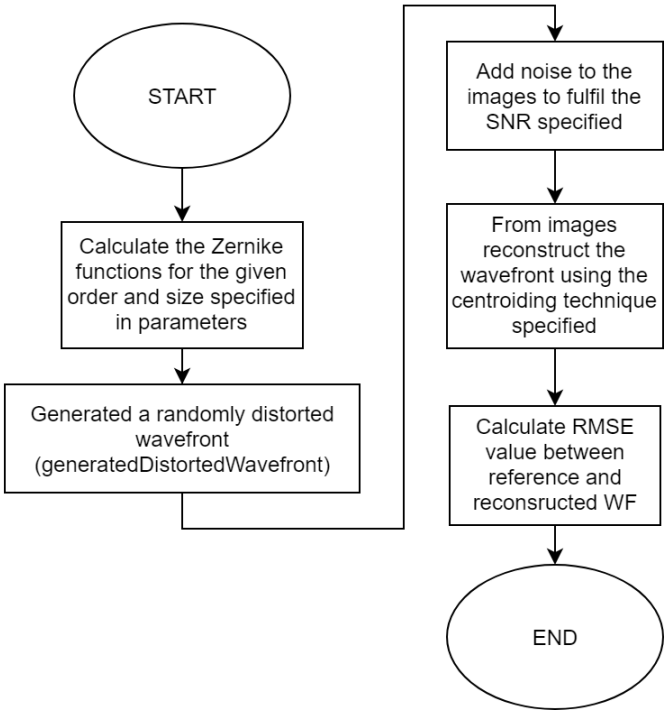


Fig-52: *Flowchart of testNoiseReductionTechnique.*

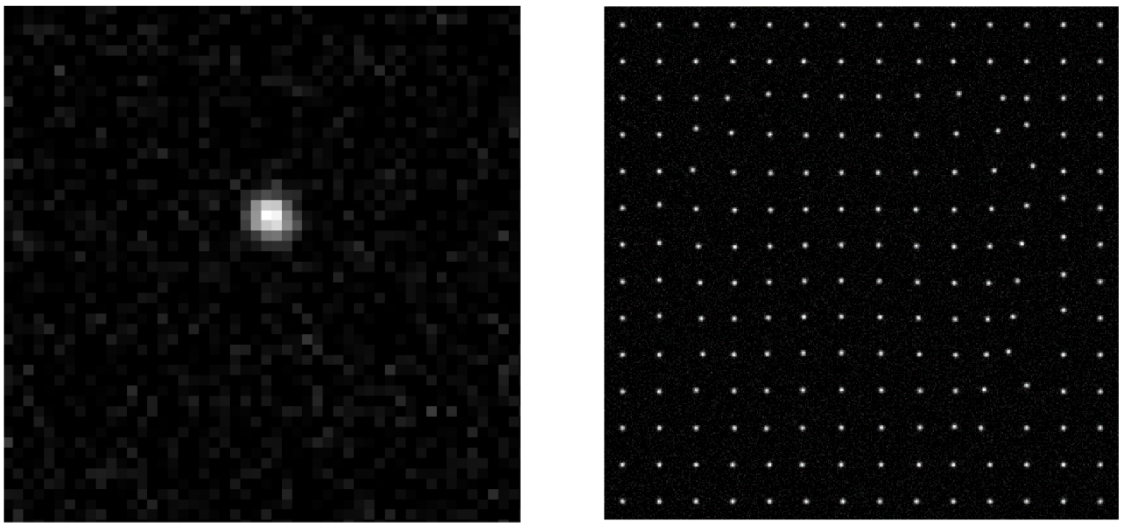


Fig-53: *Example of noise added to an image for testing a specific noise reduction technique. In the depicted example the SNR is 2.*

In case of **testRangeExtensionTechnique** the procedure is similar. For a given set of coefficients and parameters, simulated images are obtained with **getDistortedWavefront** and then the sorting algorithm to test is conducted. Then, we check if any association between reference and displaced spots is wrong and return a Boolean indicating it.

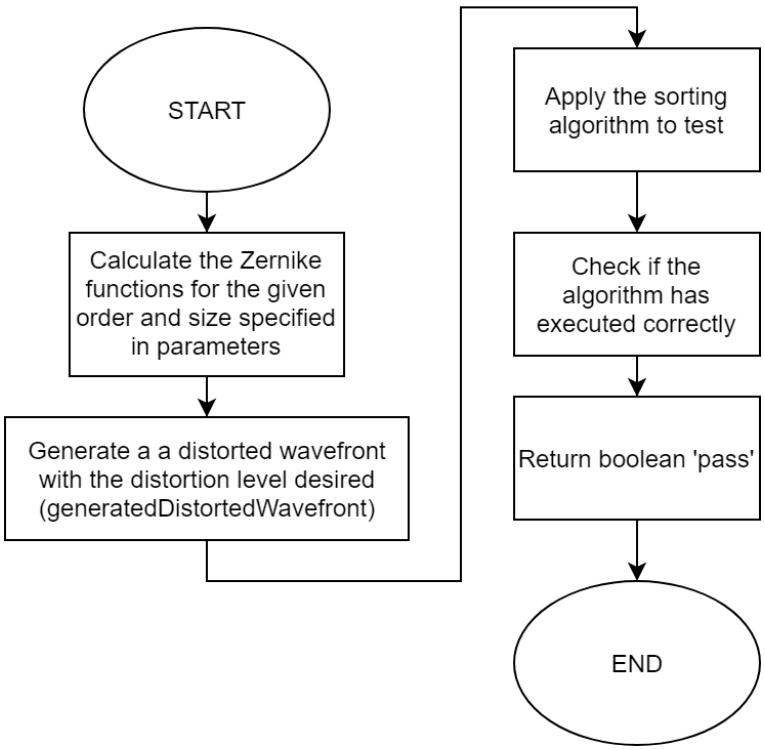


Fig-54: *Flowchart of testRangeExtensionTechnique.*

Chapter 6

Results and Discussion

In this chapter, the performance of the system implemented in MATLAB is shown and discussed. The chapter is divided in different sections where noise reduction techniques, range extension techniques and comparisons with the commercial sensor are discussed separately.

6.1 Testing Noise Reduction Techniques

By means of the simulation environment developed, the techniques for noise cancelling were tested over different conditions, in this case using **testNoiseReductionTechnique**. The first step is to model the parameters of the commercial sensor to simulate real conditions. In this way, the results can be extrapolated to give a picture of the performance of the system under real situations. Table 17 shows the parameters of the sensor according to [10] and used in the simulations. In addition, the table also displays additional parameters regarding noise cancelling techniques implemented.

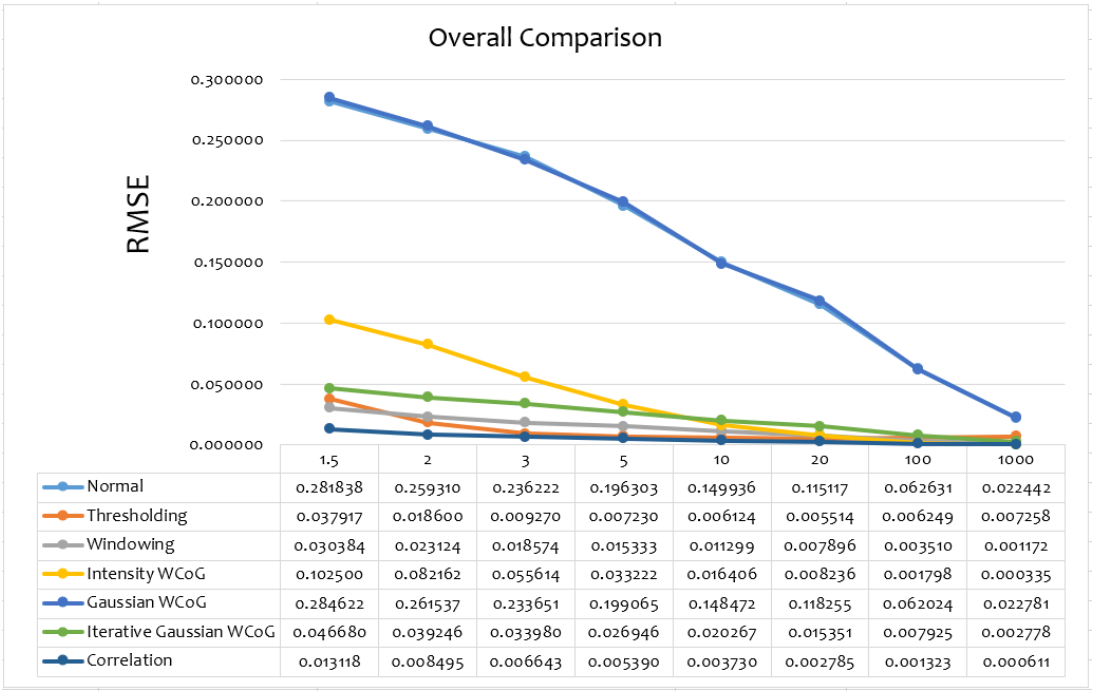
Parameter	Value
centroid_technique	Changes when testing each technique
ignore_sorting	False
sort_method	1
zonal_method	Not used
spot_px_sep	32
radius_definiton	Not used
zernikeorder	5
pixel_spacing	4.65
focal_length	5.2
lens_diameter	0.150
wavelength	0.543
input	Not used
calibration	Not used
winsize	7
inweight	2
wcog_var	4
iwcog_it	10
size_template	8
var_template	4

Table 17: *Parameters used for testing noise reduction techniques.*

6.1.1 Overall Comparison

In order to compare the implemented methods, our simulation consisted on 100 tests conducted per technique and per SNR value. Gaussian spots were of variance 1.5, and the size of the simulated grid was 14x14. It is important to recall that noise added was modelled as gaussian, as discussed in 4.1.

Figure 55 shows the average RMSE obtained from these simulations. In this figure, the adaptive-correlation implementation is not shown, as will be discussed later in 6.1.2. The SNR values for which the simulations were conducted are: 1.5, 2, 3, 5, 10, 20, 100 and 1000.



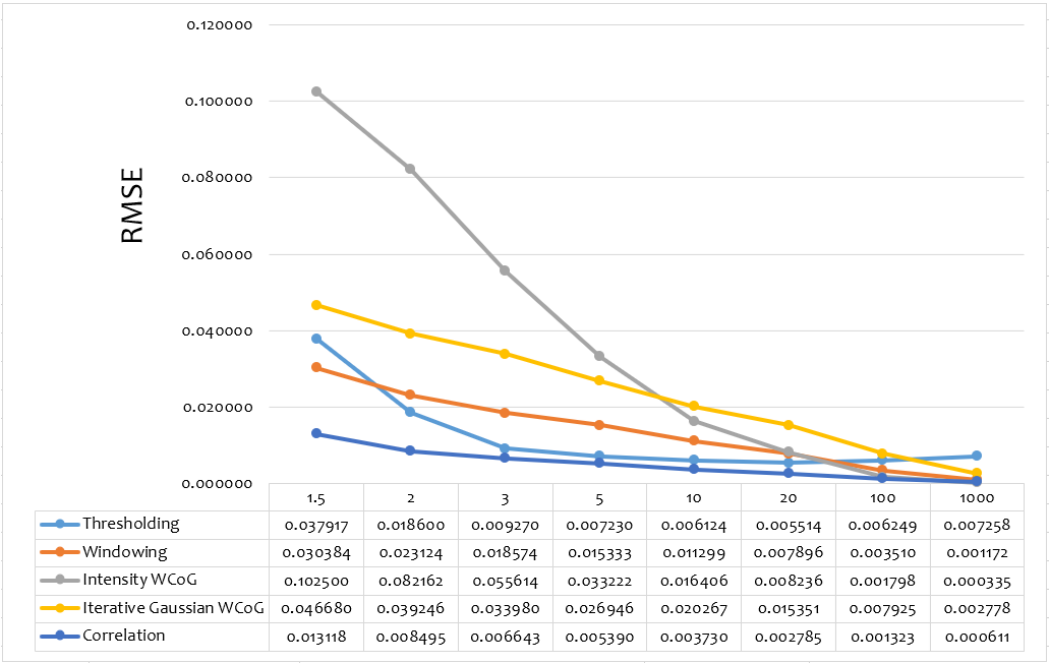
	1.5	2	3	5	10	20	100	1000
Normal	0.281838	0.259310	0.236222	0.196303	0.149936	0.115117	0.062631	0.022442
Thresholding	0.037917	0.018600	0.009270	0.007230	0.006124	0.005514	0.006249	0.007258
Windowing	0.030384	0.023124	0.018574	0.015333	0.011299	0.007896	0.003510	0.001172
Intensity WCoG	0.102500	0.082162	0.055614	0.033222	0.016406	0.008236	0.001798	0.000335
Gaussian WCoG	0.284622	0.261537	0.233651	0.199065	0.148472	0.118255	0.062024	0.022781
Iterative Gaussian WCoG	0.046680	0.039246	0.033980	0.026946	0.020267	0.015351	0.007925	0.002778
Correlation	0.013118	0.008495	0.006643	0.005390	0.003730	0.002785	0.001323	0.000611

Fig-55: *Comparison between noise cancelling techniques. The result is given as RMSE value after reconstructing the wavefront with modal approach. Signal-to-noise ratios tested: 1.5,2,3,5,10,20,100,1000.*

It is very important to remark that the purpose of this thesis is NOT to optimize every technique used. These simulations were conducted with non-optimal parameters; thus, the conclusions must be drawn with caution. As Thomas [35] pointed out, there is no clear recipe for implementing of the different centroiding schemes. The purpose of this thesis is to give the user the possibility to choose and adjust any desired technique and improve the overall performance by focusing on the correlation technique which is not implemented in the commercial sensor software. A closer look of figure 55 can be found on figure 56.

To put these results into perspective, it is important to identify the RMSE when the image is noise-free. As discussed in 3.2, there is not an error free reconstruction method, so even for no noise, the estimation of the wavefront carries some error. In case of the system implemented, for the modal algorithm, the simulations show an error in the order of 10^{-5}. That said, the following conclusions can be drawn:

- All the techniques are performing better than regular centroiding. For low SNR levels the difference is substantial, whereas for high SNR (1000), normal centroiding performs closer to the other techniques as expected.
- Gaussian WCoG exhibits very low improvement compared to normal centroiding. However, this can be explained by the use of non-optimum parameters. On the other hand, Iterative Gaussian WCoG showed a huge improvement from non-iterative. This improvement may have been higher if the number of iterations was correctly adjusted.
- Although correlation method wasn't optimized, it exhibits the best performance in terms of lower RMSE value at all SNR but 1000, where Intensity WCoG is slightly higher. In this sense, it seems that Intensity WCoG is optimal for high SNR values. This also depends on the weight of the intensity function, in this case a factor 2.
- Between SNR values of 20 and 100, thresholding seems to saturate and the RMSE starts increasing, although this increase is relatively low. For SNR between 2 and 20 is the closest technique to correlation-based centroiding. Again, this may change if the rest of the techniques are optimized.

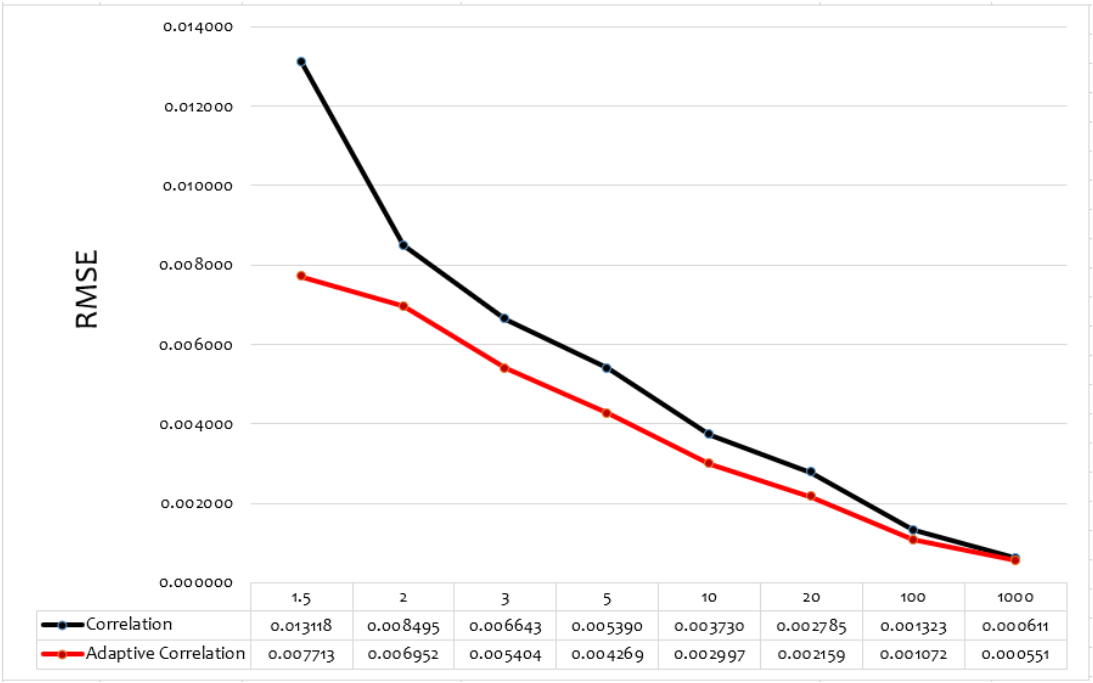


	1.5	2	3	5	10	20	100	1000
Thresholding	0.037917	0.018600	0.009270	0.007230	0.006124	0.005514	0.006249	0.007258
Windowing	0.030384	0.023124	0.018574	0.015333	0.011299	0.007896	0.003510	0.001172
Intensity WCoG	0.102500	0.082162	0.055614	0.033222	0.016406	0.008236	0.001798	0.000335
Iterative Gaussian WCoG	0.046680	0.039246	0.033980	0.026946	0.020267	0.015351	0.007925	0.002778
Correlation	0.013118	0.008495	0.006643	0.005390	0.003730	0.002785	0.001323	0.000611

Fig-56: *Closer look of figure 54, without normal centroiding and Gaussian WCoG.*

6.1.2 Adaptive-Correlation

Same simulation procedure was conducted to test the performance of the adaptive-correlation technique that was presented in 5.3. For this test, the variance of the template was calibrated, starting from the default value of 4, as shown is Table 17. The outcome of the simulations is depicted in figure 57.



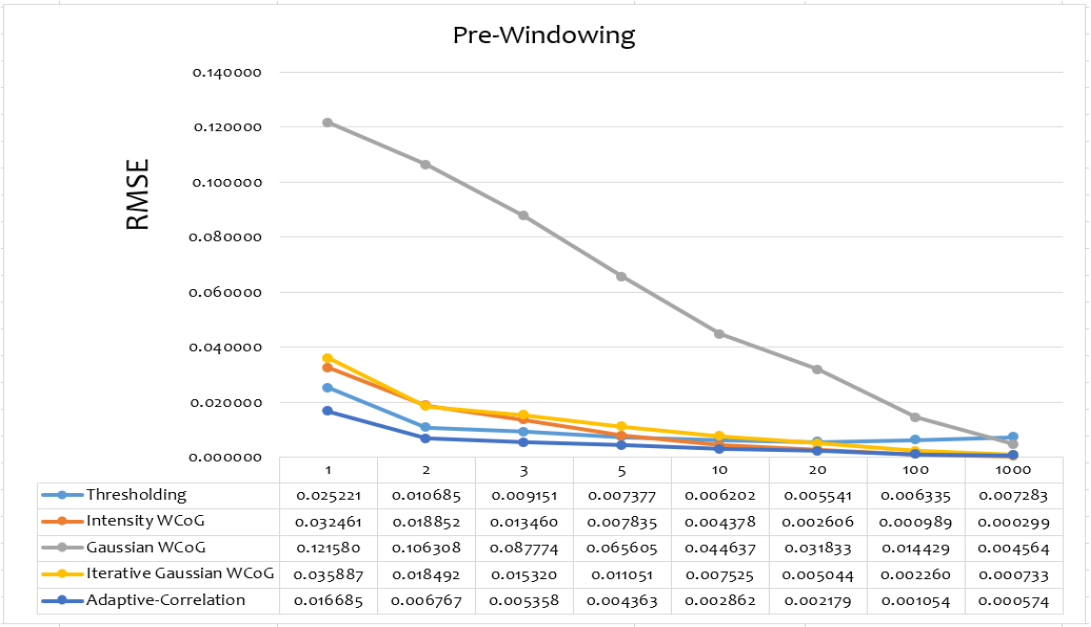
	1.5	2	3	5	10	20	100	1000
Correlation	0.013118	0.008495	0.006643	0.005390	0.003730	0.002785	0.001323	0.000611
Adaptive Correlation	0.007713	0.006952	0.005404	0.004269	0.002997	0.002159	0.001072	0.000551

Fig-57: *Comparison of regular correlation technique vs adaptive correlation.*

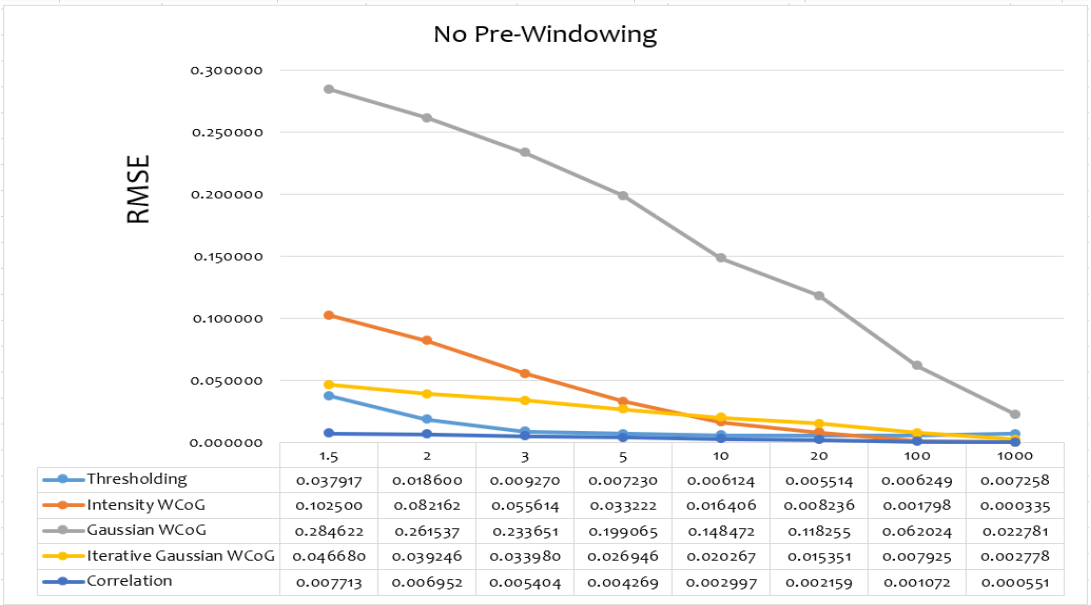
The results are overwhelming. The adaptive-correlation technique proposed here is always superior. This difference is higher when the SNR, as expected, is low and becomes less important for high SNR. In terms of computational efficiency, adaptive-correlation involves a high price. For processing a grid 14x14, regular correlation takes around 2 seconds in our system implementation, whereas the adaptive-correlation takes around 9 seconds. This occurs because in the adaptive approach cross-correlation is computed many times. This could be dramatically reduced, if the template was calibrated only for the first spot, and not for each one. However, as our system is meant to be operating offline, time consuming is not as important as accuracy. Here, is worth mentioning that the approach to achieve sub-pixel accuracy may change slightly the results. In this case, gaussian 5-point interpolation was used according to equations 4.10 and 4.11. However, there are other possible implementations, like 3-point parabola interpolation or regular centroiding [35]. It would be interesting to implement other approaches and compare their performance.

6.1.3 Pre-Windowing

These techniques are not limited to an isolated use. By combining multiple techniques, the performance can be also improved. In this section, the effect of introducing a pre-windowing is studied, before applying the techniques. The results are displayed in figure 59. Normal centroiding was excluded from the test, as that is equivalent to regular windowing. Same simulation procedure of previous sections was carried out.



	1	2	3	5	10	20	100	1000
Thresholding	0.025221	0.010685	0.009151	0.007377	0.006202	0.005541	0.006335	0.007283
Intensity WCoG	0.032461	0.018852	0.013460	0.007835	0.004378	0.002606	0.000989	0.000299
Gaussian WCoG	0.121580	0.106308	0.087774	0.065605	0.044637	0.031833	0.014429	0.004564
Iterative Gaussian WCoG	0.035887	0.018492	0.015320	0.011051	0.007525	0.005044	0.002260	0.000733
Adaptive-Correlation	0.016685	0.006767	0.005358	0.004363	0.002862	0.002179	0.001054	0.000574



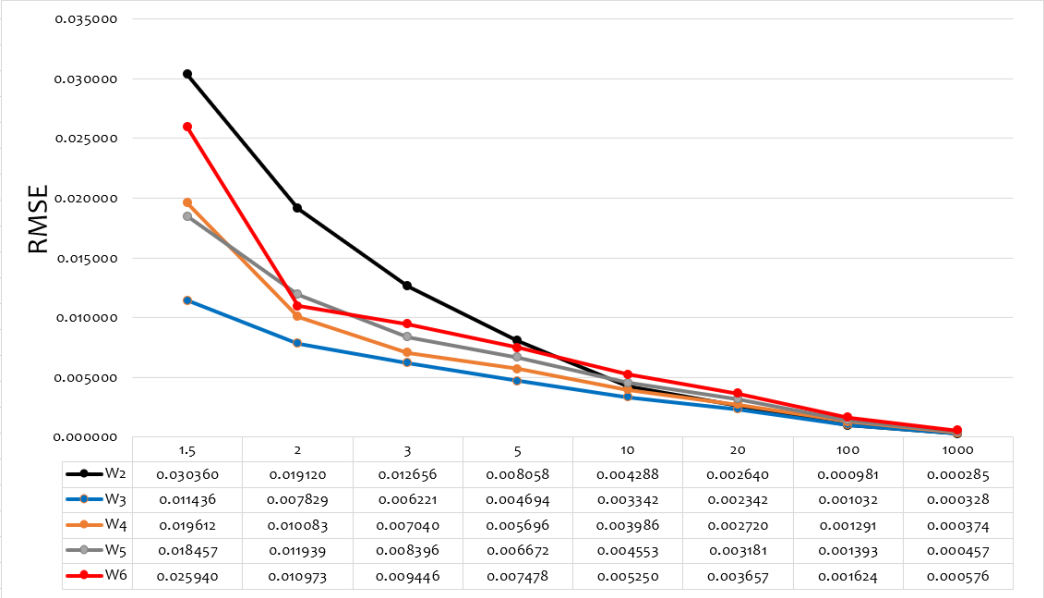
	1.5	2	3	5	10	20	100	1000
Thresholding	0.037917	0.018600	0.009270	0.007230	0.006124	0.005514	0.006249	0.007258
Intensity WCoG	0.102500	0.082162	0.055614	0.033222	0.016406	0.008236	0.001798	0.000335
Gaussian WCoG	0.284622	0.261537	0.233651	0.199065	0.148472	0.118255	0.062024	0.022781
Iterative Gaussian WCoG	0.046680	0.039246	0.033980	0.026946	0.020267	0.015351	0.007925	0.002778
Correlation	0.007713	0.006952	0.005404	0.004269	0.002997	0.002159	0.001072	0.000551

Fig-58: *Performance of applying a window before using different techniques. The results correspond to a grid 14x14 for the different SNR levels.*

The results clearly show that pre-windowing is beneficial for achieving less RMSE combined with some techniques, but not all. In case of thresholding and correlation, the RMSE achieved is similar or even worst at low SNR levels. However, for Intensity WCoG, Gaussian WCoG and Iterative Gaussian WCoG, one finds an improvement. We can conclude that pre-windowing is useful when using WCoG methods. Further research should be done in order to determine the benefits of combining other techniques.

6.1.4 Intensity Weight Comparison

Finally, different weights for the intensity WCoG method are studied. As shown in table 17, the default value of this technique is 2, but this is not restricted to just this value. Some values may be more effective for different SNR levels. This is depicted on figure 59. Again, same simulation procedure as in previous sections is used.



	1.5	2	3	5	10	20	100	1000
W2	0.030360	0.019120	0.012656	0.008058	0.004288	0.002640	0.000981	0.000285
W3	0.011436	0.007829	0.006221	0.004694	0.003342	0.002342	0.001032	0.000328
W4	0.019612	0.010083	0.007040	0.005696	0.003986	0.002720	0.001291	0.000374
W5	0.018457	0.011939	0.008396	0.006672	0.004553	0.003181	0.001393	0.000457
W6	0.025940	0.010973	0.009446	0.007478	0.005250	0.003657	0.001624	0.000576

Fig-59: *Comparison of different weight for the Intensity WCoG method.*

As expected, the weights exhibit different behaviors for different SNR levels. For instance, the weight 2 is optimum for high SNR (up to 100) and, on the other hand, all the other weights are superior until SNR goes up to 10. However, is interesting to appreciate that for the rest of SNRs the winner is clearly weight 3. In none of the SNR levels any other weight is performing better and weight 3 or 2, therefore they may not be used. From the results a clear recipe emerges: use weight 2 for high SNR and use weight 3 for low to high SNR.

6.2 Testing Dynamic Range Increasing Techniques

In this section, the performance of the implemented Zernike-based sorting algorithm for range extension is presented, compared to the conventional algorithm and sorting by minimum Euclidean distance. Results are depicted in figure 60.

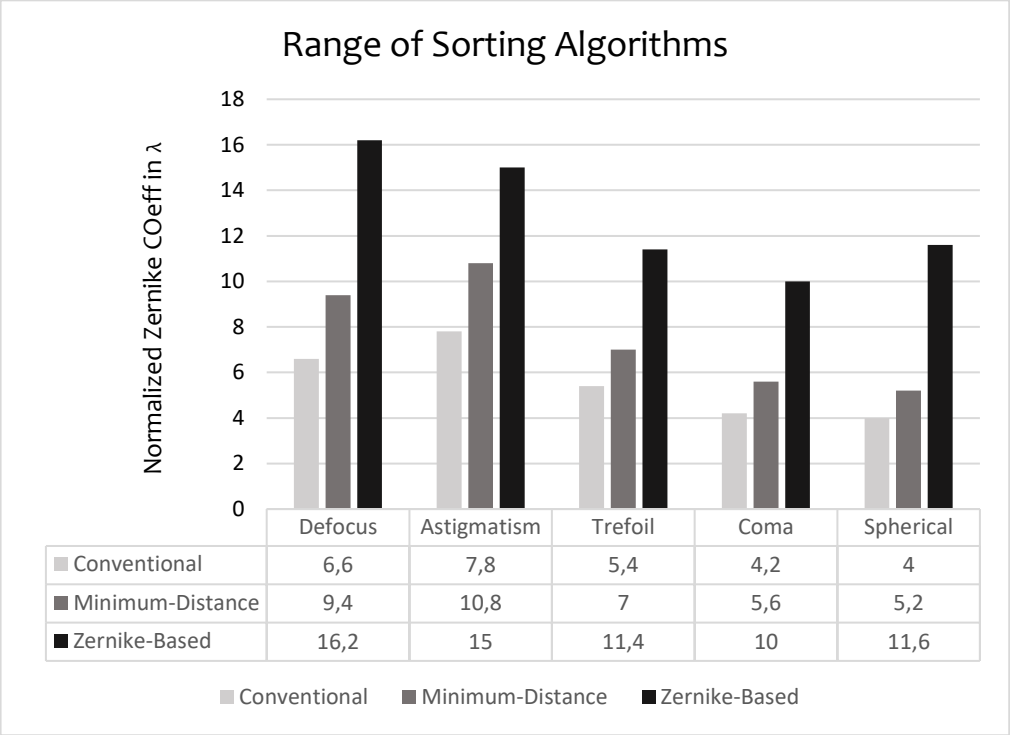


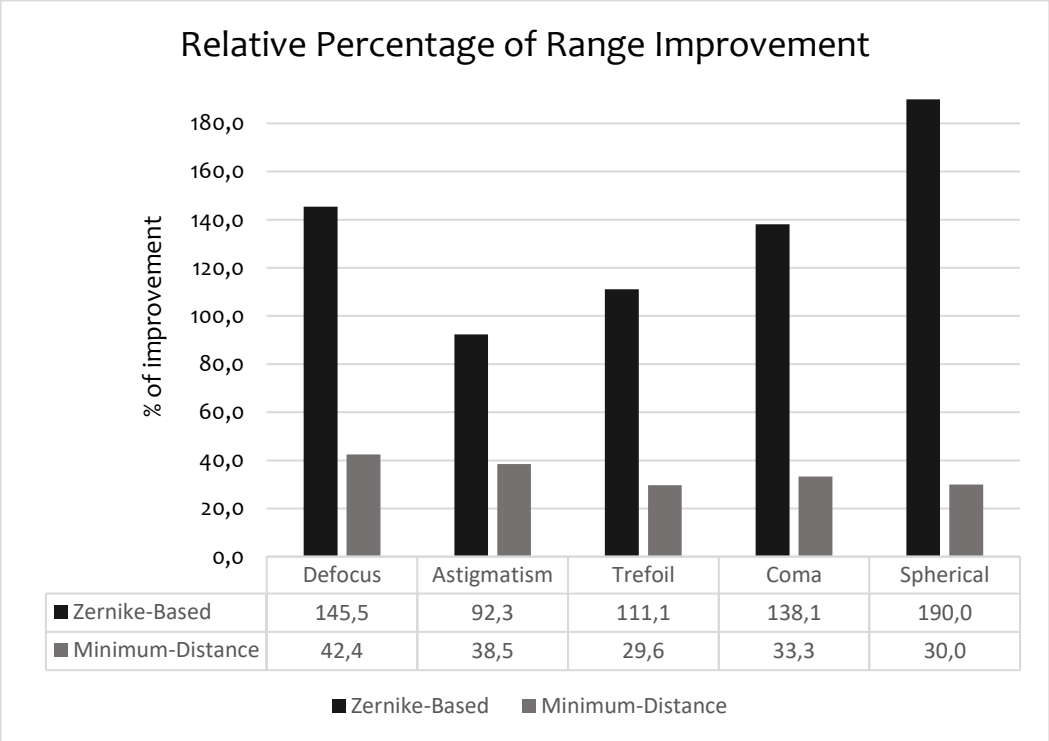
Fig-60: *Range of the algorithms implemented in the system. The values are obtained for positive coefficients. Unit are in wavelengths, using in this case 543nm. Results obtained for a grid 14x14.*

These results have been obtained with the simulation toolbox developed, using the parameters that models the characteristics of the commercial sensor (Table 17). The grid generated was 14x14 and the simulated light spots were of variance 1.5. No noise was added to the simulated sensor images and modal approach was used for reconstruction. The interval between tested coefficients in λ was of 0.2. The test was conducted for the following aberrations: defocus, astigmatism, trefoil, coma and spherical.

To determine if the algorithm has been able to resolve correctly spot association, it is important to remark that the criteria for the conventional algorithm is slightly different. In the conventional case, the spots are not being sorted between reference and displaced groups. On the contrary, we are running over the static grid and computing each centroid within a cropped image corresponding to each grid sub-area. This implies that if a spot starts to be close to the border of the sub-area, or other external spot starts coming in, the centroid position is not going to be accurate and there will be some error in the reconstruction.

As we said, the RMSE from the wavefront reconstruction, when there is no noise influence, is in the order of 10^{-5} for the modal approach. Therefore, we fixed the criteria to state that the conventional algorithm has failed, when RMSE reaches a value of 10^{-3}. In case of the sorting algorithms, if the sorting has been conducted correctly, we assure that the error is within minimum levels, so we don't need to check for the RMSE.

In order to interpret the results, it is indispensable to give a measure of relative improvement. Figure 61 shows the relative percentage increase in relation to the conventional algorithm. We see how the Zernike-based algorithm is increasing the range between 111,1 and 190 percentual points. In case of sorting with minimum distance, the increase is significantly lower, between 30 and 42,4 percentual points. This second algorithm is helping in situations where displaced spots start leaving their corresponding sub-areas, or other external spots are entering other sub-areas, but its performance is drastically limited compared to the Zernike-based algorithm.



	Defocus	Astigmatism	Trefoil	Coma	Spherical
Zernike-Based	145,5	92,3	111,1	138,1	190,0
Minimum-Distance	42,4	38,5	29,6	33,3	30,0

Fig-61: *Relative percentage of range improvement for the minimum-distance and Zernike-based algorithm. Results obtained for a grid 14x14.*

Having said that, the results demonstrate great performance of the technique proposed in [42]. The implemented Zernike-based sorting algorithm extends the range extensively compared to the other approaches. Moreover, in some cases, like for SA, we find that the overlapping of two displaced spots is limiting before sorting, because the algorithm of detecting spots is failing first. An example is depicted in figure 62. Notice that for negative coefficients the results are the same, except in case of SA in the Zernike approach, because the limiting factor is the overlapping of displaced spots.

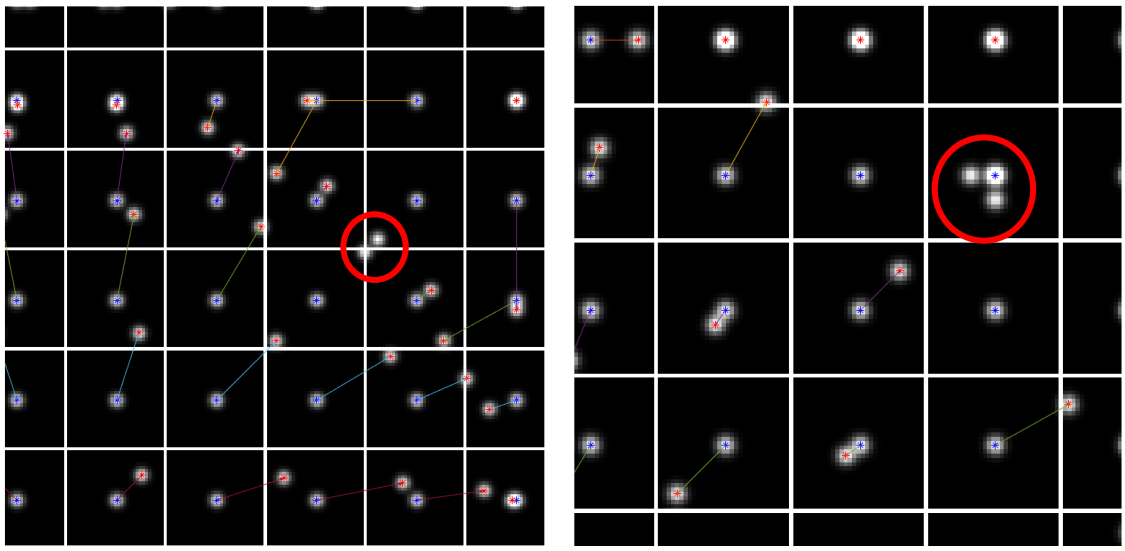


Fig- 62 *Example of limited performance due to the overlapping of displaced spots. In the left, SA of 11.7 waves. In the right, SA of -8.8 waves. Indicated with the red circles are displaced spots that are overlapping and therefore the detecting algorithm is failing before applying sorting algorithms.*

Finally, the processing time has been studied for the algorithms of minimum distance and Zernike-based. For the minimum distance sorting, time consumed is in between 8-10ms, whereas the Zernike-based algorithm implemented take about 100ms (results for a grid 14x14). Therefore, we conclude that the second algorithm is dramatically less efficient compared to minimum distance sorting. However, two things must be considered here. First, the algorithm is susceptible for re-coding and obtaining much better efficiency, and second, the system developed is meant to be an offline system, so time efficiency is always of less relevance than accuracy or range.

6.3 Real Wavefront Reconstruction

In this final section, the system developed is tested with real wavefront images, and results are compared with the standard software of the commercial sensor. Two images representing two distorted wavefronts were reconstructed and analyzed. First, modal and zonal approach will be compared and then the results of the commercial sensor will be contrasted with the system developed in this work.

6.3.1 Zonal vs Modal

In order to compare zonal vs modal reconstruction, the first wavefront image was selected. For the zonal approach, least-squares solution was adopted with Southwell Geometry. It is important to remark that the unit circle definition was done as depicted in figure 63 for a grid 14x14. Parameters of the sensor were modelled according to Table 17.

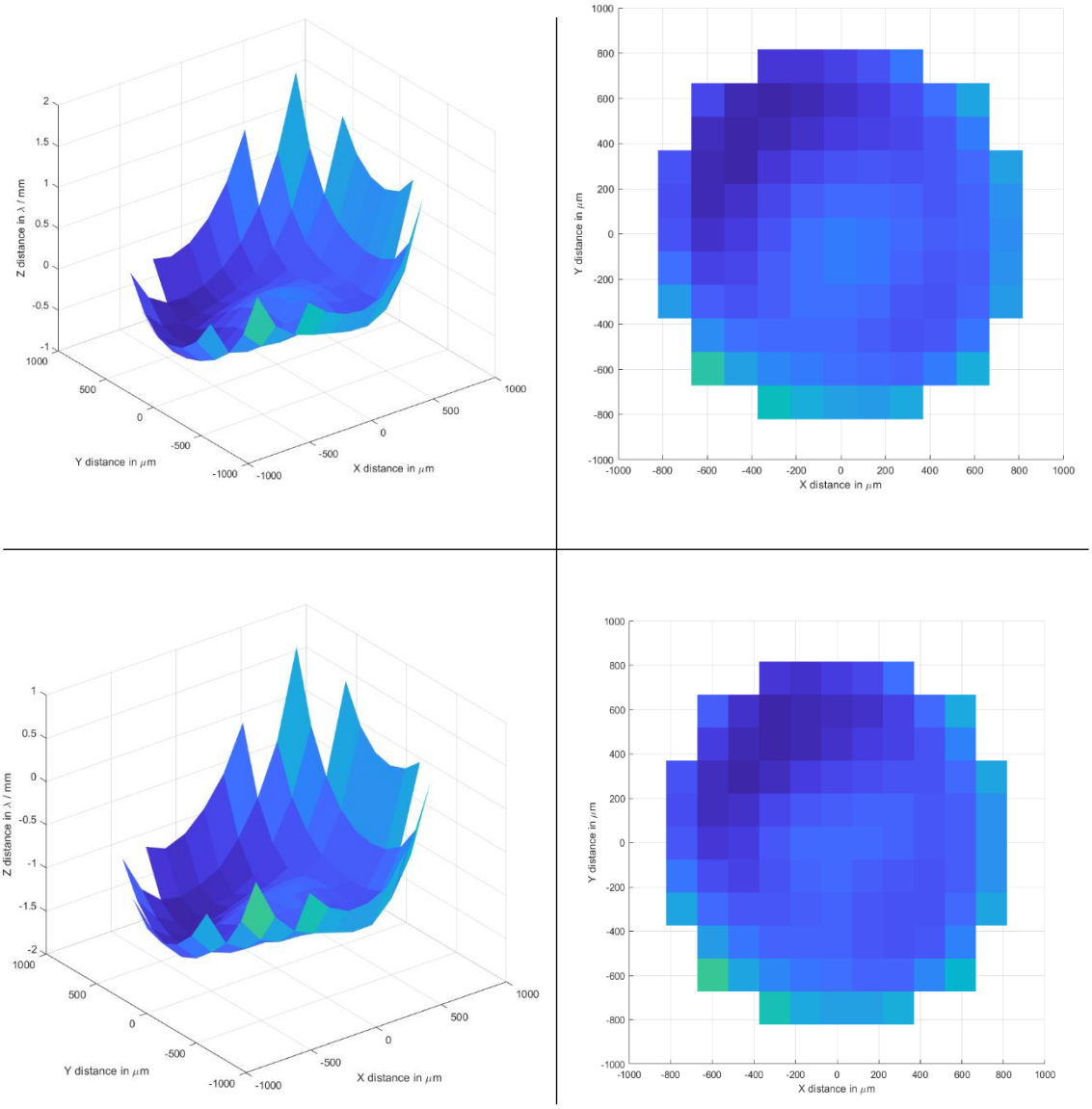
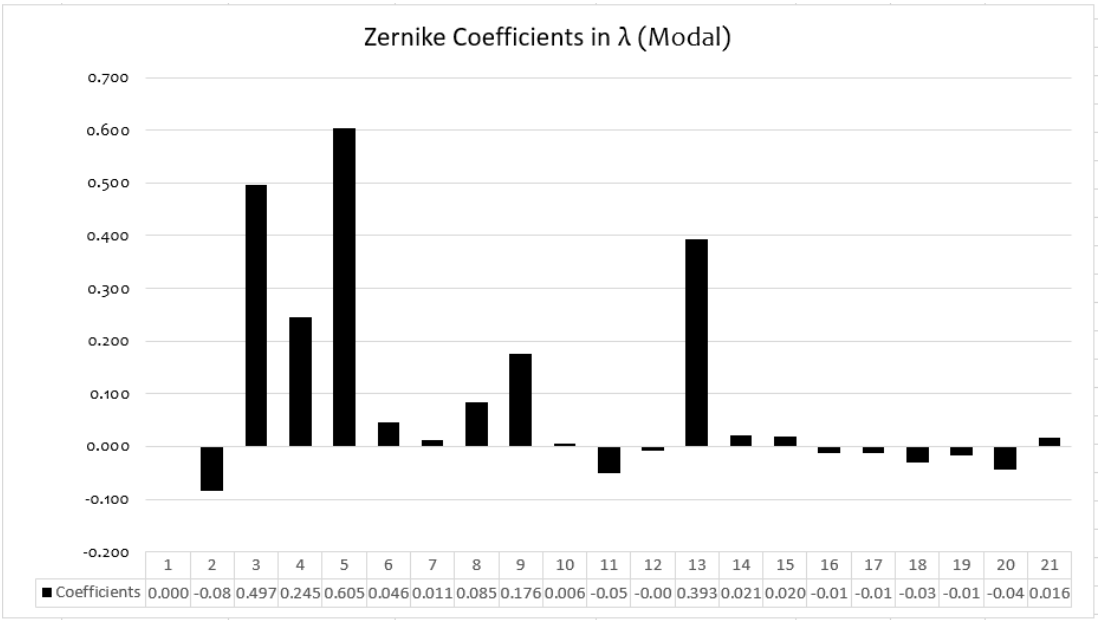


Fig-63: *Comparison of the wavefront 3D and 2D shapes obtained from the modal and zonal approach. Top: Modal. Bottom: Zonal.*



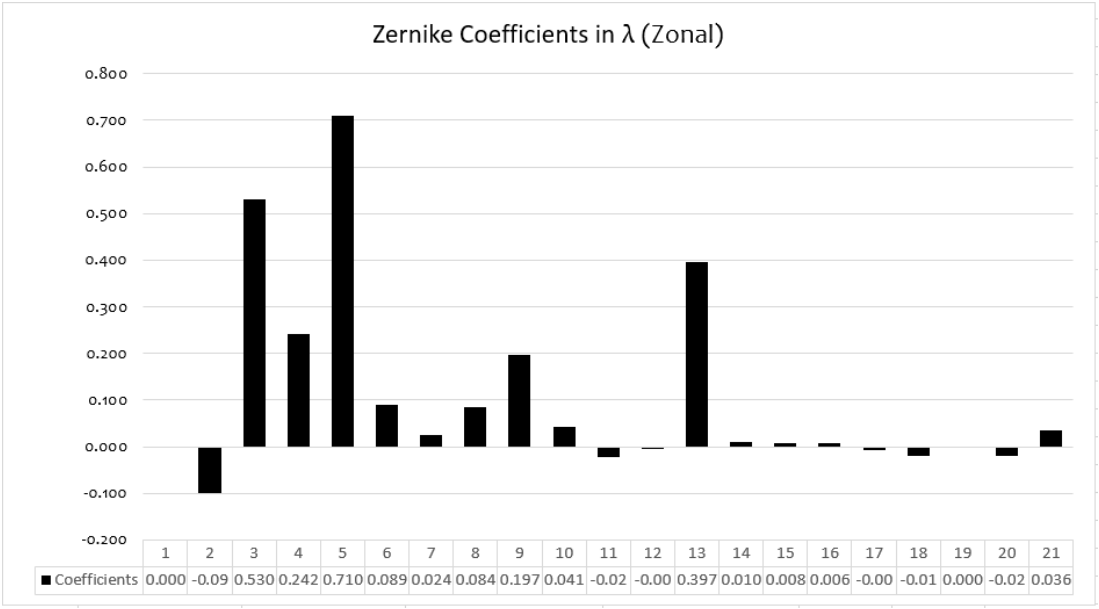


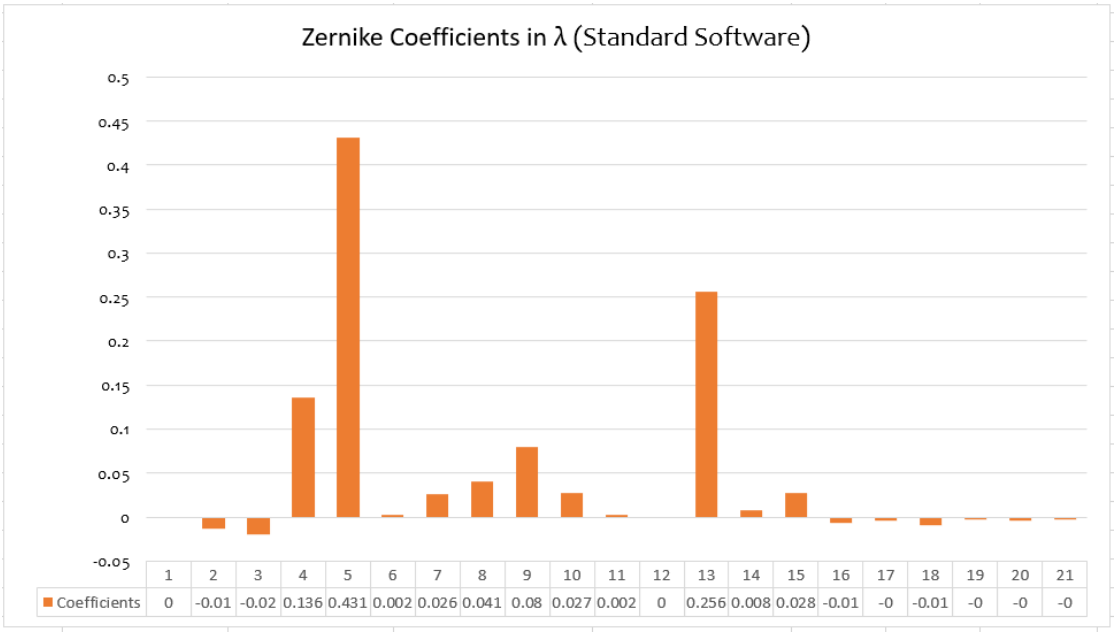
Fig-64: *Comparison of the Zernike coefficients from modal and zonal reconstruction. Coefficients are in units of wavelengths and calculated up to the 5th order. The indexing uses is OSA/ANSI (equation 2.18).*

Comparing the results, the coefficients exhibit different values. This is caused because both approaches are intrinsically different. As discussed in Chapter 5, in zonal reconstruction algorithms, every grid value influences the wavefront estimation, because least-squares solution considers the full grid for reconstruction. Missing slope-measurements on the grid were extrapolated outside the pupil domain. On the other hand, modal reconstruction only considers values that are part of the unit circle defined. Despite the mismatch, the difference is relatively low, especially considering relevant coefficients like astigmatism, coma or spherical aberration.

6.3.2 Comparing the Standard Software

Finally, the system is tested and compared with the standard software of the sensor. The results are shown for both wavefront samples. Regarding method of reconstruction, modal approach was used for the results corresponding to the system developed in this thesis.

WAVEFRONT 1 (d45.bmp)



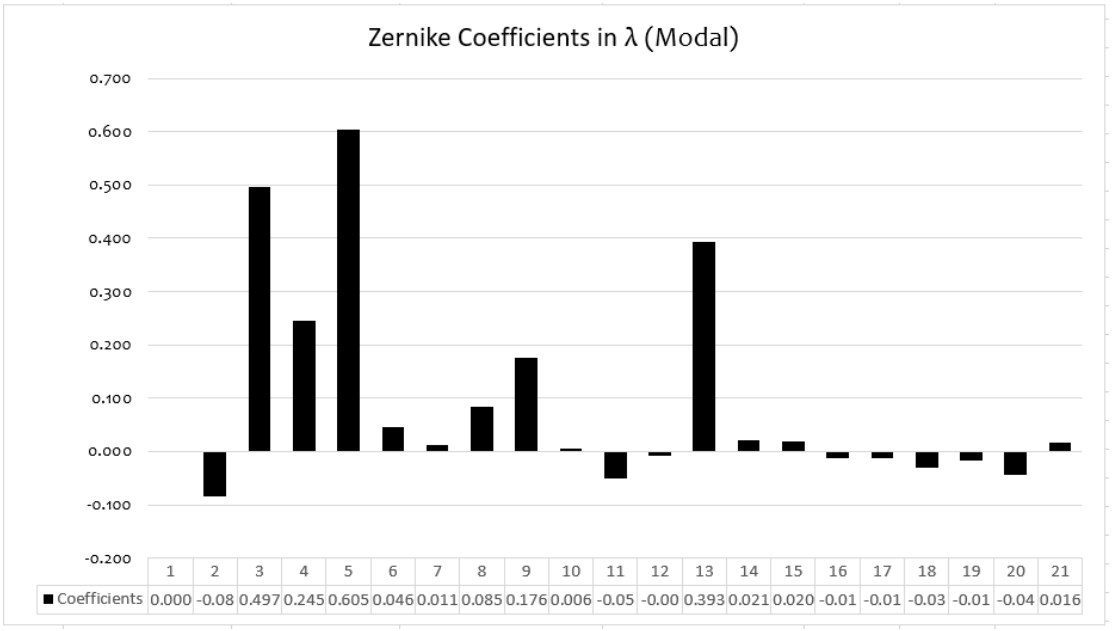


Fig-65: *Comparison of the first 21 Zernike Coefficients obtained from the standard software and the system developed in this thesis (image 'd45.bmp'). Units are in waves. Notice that the first 3 coefficients must be ignored.*

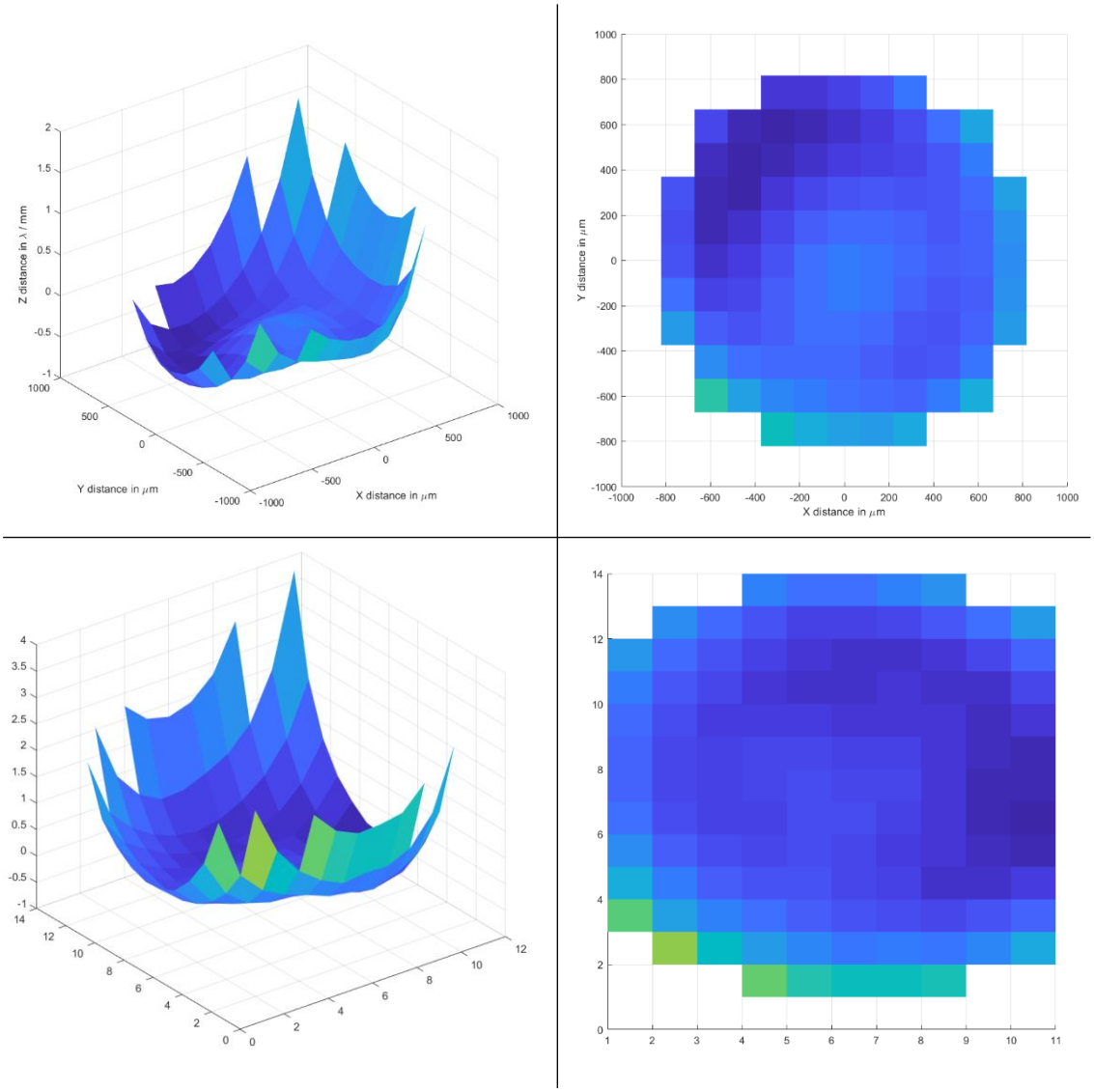


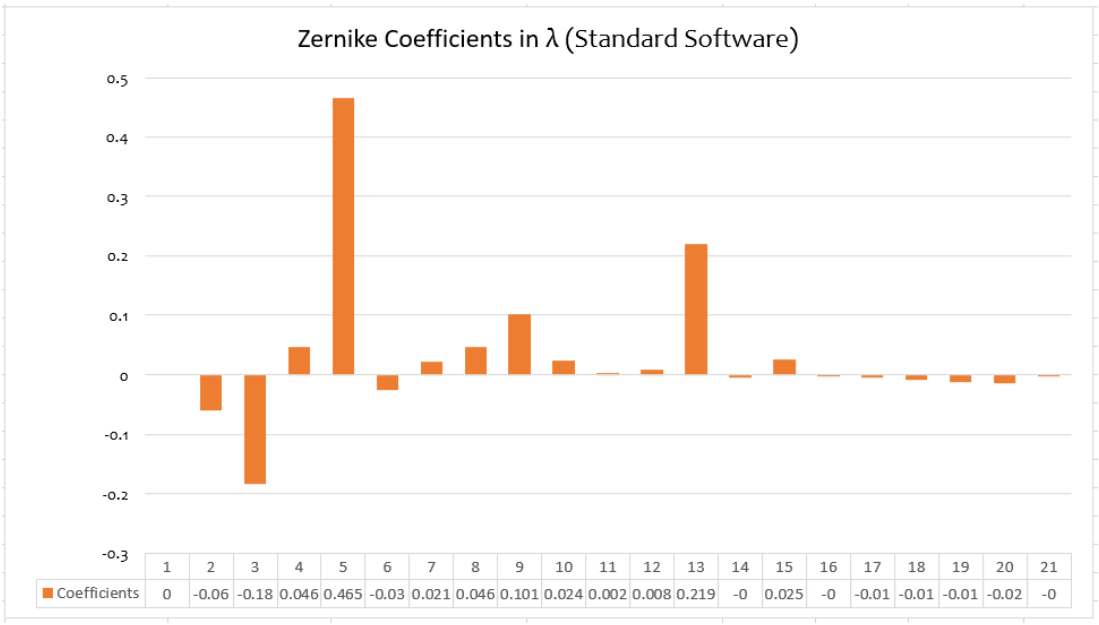
Fig-66: *Comparison of the wavefront 3D and 2D shapes obtained from the standard software and the system developed in this thesis ('d45.bmp'). Top: System developed. Bottom: Standard Software.*

In figure 65, a great contrast can be seen between coefficients obtained. Nonetheless, is important to notice that the relative contribution of each coefficient is similar. In both solutions, most relevant coefficients are defocus (5), spherical (13), oblique astigmatism (4) and vertical coma (9). Also notice that the first three coefficients must be ignored, as piston is not determined for the modal approach, and tilt and tip information is lost due the misalignment fix.

Many factors can influence the results. First, there are multiple normalizations that can be used for the Zernike polynomials and for the wavefront. If any factor used is different, this will lead to mismatch in the coefficients estimation. In addition, the area/spots considered for the reconstruction can change dramatically the result. In this sense, we see that both solutions consider different set of points for reconstruction. In the system developed, the grid was 14x14, whereas in the standard software, the wavefront is reconstructed over a grid 14x11. This discrepancy causes both solutions to differ greatly from each other, independently

from different normalization factors used. The shape of the wavefronts depicted in figure 66 confirms this mismatch. Is worth mentioning that the standard software is considering non-symmetrical grids, whereas in our system the wavefront is always reconstructed within a symmetrical grid.

WAVEFRONT 2 (c4.bmp)



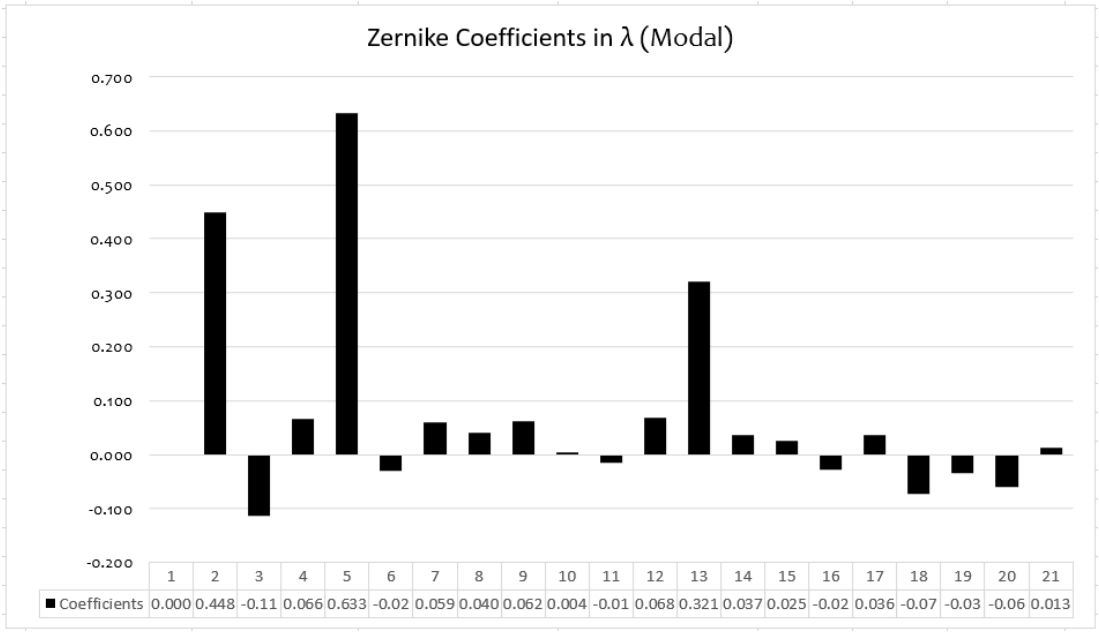


Fig-67: *Comparison of the first 21 Zernike Coefficients obtained from the standard software and the system developed in this thesis (image 'c4.bmp'). Units are in waves. Notice that the first 3 coefficients must be ignored.*

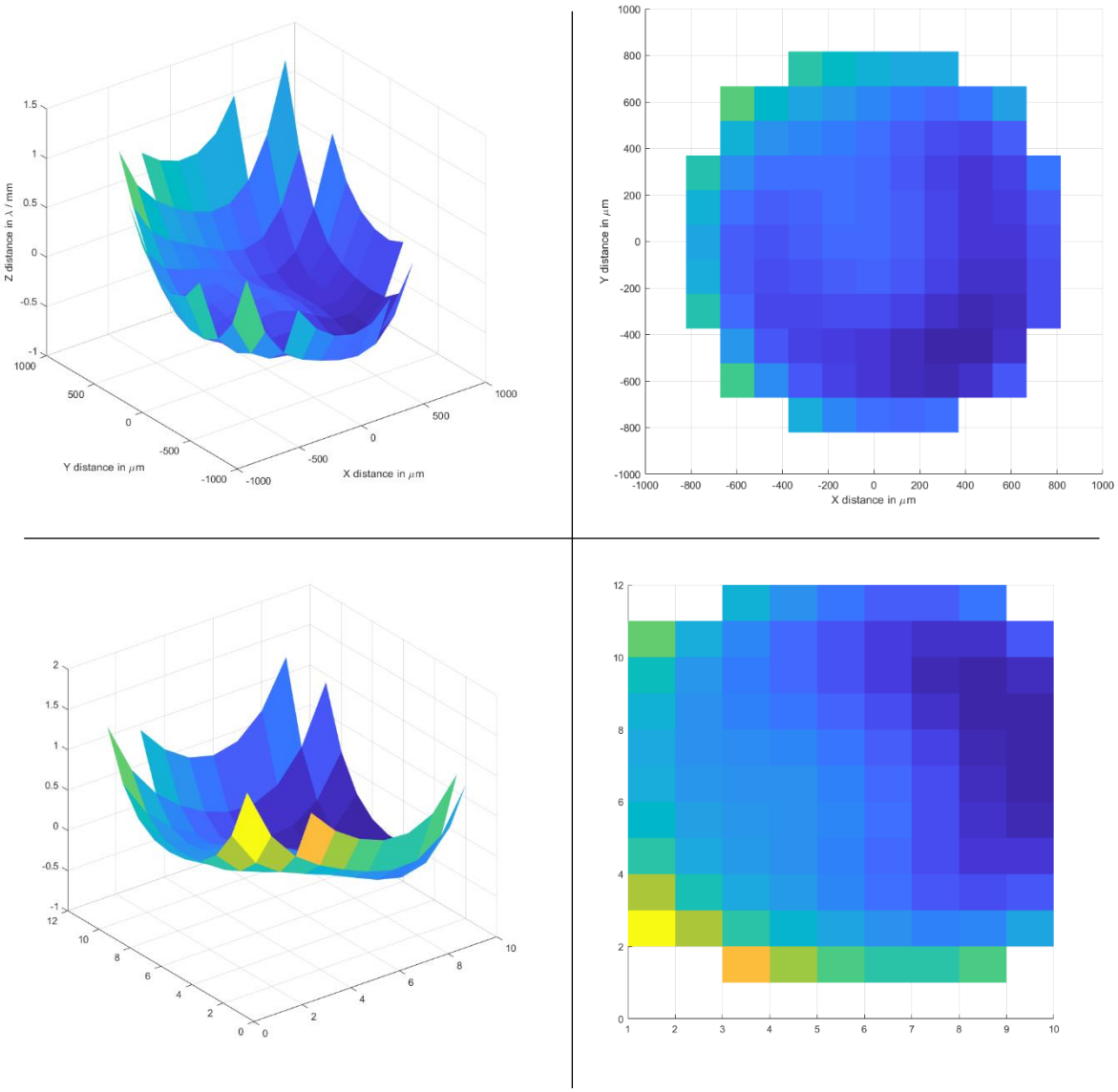


Fig-68: *Comparison of the wavefront 3D and 2D shapes obtained from the standard software and the system developed in this thesis ('c4.bmp'). Top: System developed. Bottom: Standard Software.*

This second comparison confirms the findings discussed with the first wavefront analyzed. The coefficients are again not matching although the relative contribution is similar, being in this case the most relevant coefficient defocus (5) and spherical (13). Again, the grid considered in the reconstruction is different: 14x14 and 12x10 for the system developed and the standard software respectively. These results show the importance that the definition of the radius, as well as the set of points considered, have on the reconstructed wavefront.

Chapter 7

Conclusions

In this thesis, a full system of characterization of optical aberrations from SH sensor measurements has been successfully developed in MATLAB. Multiple techniques have been implemented to improve the performance by reducing noise and extending the dynamic range of the sensor. In this sense, a basic technique has been proposed and tested for optimizing correlation-based centroiding. In addition, a basic simulation environment has been developed to test the performance of these techniques under multiple SNR conditions. Finally, the system developed has been compared with the standard software of the commercial sensor. The three objectives of this thesis (1.2) have been successfully fulfilled.

The comparison between the noise cancelling methods has confirmed the difficulty and importance of selecting optimal parameters, and the challenge of choosing the optimum technique for each situation. However, correlation-based centroiding has been proven to be the less sensitive to noise and closer to optimum as other authors stated. It Is remarkable the performance demonstrated by Intensity WCoG, being close to correlation, or even superior over high SNR with the correct adjustment of the weight. In case the optimized correlation-based centroiding (adaptive correlation), the results have shown improvement on spot location estimation, compared to regular correlation, as expected. However, the technique is much less efficient, and therefore is discarded for online systems the way it is implemented in this system. The range extension methods have been successfully implemented and tested. Zernike-based sorting has demonstrated very high range extension, being in cases like SA not the limiting factor, as the overlapping of displaced spots make the system fail first.

The standard software exhibited different results than the presented system for the two sensor images studied. As discussed, the mismatch between results are caused by different normalization factors, but most importantly, by considering different set of spots and different radius definition. Although the relative contribution of the coefficients was similar, these results highlight the importance and effect of the domain considered for wavefront reconstruction.

In conclusion, the system developed is flexible, implements a nice variety of options to adapt for multiple situation and provides the tools for testing and conducting research on the performance of SH sensors. We can state that it offers a solid alternative to the commercial sensor WFS-150-7AR from ThorLabs. However, some limitations can be identified, and further research and work should be carried out. The future work to be done can be summarized in:

- **Developing a more complete simulation environment**: In terms of limitations, we can say that the main limitations are found in the simulation environment. First, the test was conducted for unique light spot pattern (gaussian with specific variance), and second, effects like diffraction or number of photons were not considered. This simulation environment is valid for basic tests but is lacking a better modelling of the sensor physics.

- **Improving Efficiency:** The efficiency of the system in terms of computational cost has been demonstrated to be low, taking around 2 seconds for reconstructing the wavefront over a grid 14x14. In the context of this thesis, this is not of much relevance as the system is meant to be an offline system. However, if this system were desired to be applied in such fields as AO, big efficiency improvement must be conducted.

- **Implementing new wavefront reconstruction methods**: This thesis has focused on the additional techniques for reducing noise impact on the measurement of spot locations. Nevertheless, there are many approaches for wavefront reconstruction rather than classical zonal and modal algorithms, that haven't been implemented. The next step would be to implement state-of-the-art wavefront reconstruction methods.

- **Researching on optimization of noise cancelling techniques**: As discussed, there is no clear recipe for selecting optimal parameters for the noise cancelling techniques discussed here. This highlights the need for further research on how to adjust the parameters of these techniques for each noise level scenario. In this sense, it will be of special interesting to test other approaches for sub-pixel accuracy in correlation technique, and also, to study the benefits of combining different techniques.

Bibliography

[1] L. Traxler, “Testing the optical quality of intraocular lenses regarding postoperative tilt and decentration”, Ph.D. dissertation, Faculty of Physics, Technische Universität Wien, Austria, 2018

[2] “Cataract Statistics & Resources,” Laser Eye Surgery Hub, 13-Apr-2019. [Online]. Available: https://www.lasereyesurgeryhub.co.uk/cataract-statistics/. [Accessed: 17-Apr-2019].

[3] “Cataract surgery: how countries compare,” Cataract surgery: how countries compare - Product - Eurostat. [Online]. Available: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20190108-1. [Accessed: 31-May-2019].

[4] R. Tyson, Principles of Adaptive Optics, Third Edition. CRC Press, 2010.

[5] Chanan, G. Principles of wavefront sensing and reconstruction. Proceedings: Summer School on Adaptive Optics; Center for Adaptive Optics (CfAO): Santa Cruz, CA, USA, p. 5, 2000.

[6] M. Saleh, “Optique adaptative pour l’ophtalmologie,” Journal Français d’Ophtalmologie, vol. 39, no. 4, pp. 380–386, Apr. 2016.

[7] H. I. Campbell and A. H. Greenaway, “Wavefront Sensing: From Historical Roots to the State-of-the-Art,” EAS Publications Series, vol. 22, pp. 165–185, 2006.

[8] “Adaptive Optics - Deformable Mirrors,” Applications - Adaptive Optics - Deformable Mirrors | ALPAO.com. [Online]. Available: https://www.alpao.com/adaptive-optics/alpao-applications.html. [Accessed: 17-Apr-2019].

[9] W. H. Southwell, “Wave-front estimation from wave-front slope measurements,” Journal of the Optical Society of America, vol. 70, no. 8, p. 998, Aug. 1980.

[10] Shack-Hartmann Wavefront Sensors – WFS150-7AR., THORLABS. New Jersey., USA. Jul 2018. Available: https://www.thorlabs.com/catalogpages/Obsolete/2018/WFS150-7AR.pdf [Accessed: 10-Apr-2019].

[11] B. E. A. Saleh and M. C. Teich, “Fundamentals of Photonics,” Wiley Series in Pure and Applied Optics. John Wiley & Sons, Inc., 14-Aug-1991.

[12] Kuria, J. M., Schon, R., & Börret, R. “A flatbed scanner based wavefront sensing unit for optics quality control. In Proceedings”. 18th World Conference on Nondestructive Testing. Apr. 2012.

[13] “Introduction to Adaptive Optics and Deformable Mirrors,” Edmund Optics Worldwide. [Online]. Available: https://www.edmundoptics.jp/resources/application-notes/optics/introduction-to-adaptive-optics-and-deformable-mirrors/. [Accessed: 18-Apr-2019].

[14] E. L. Dereniak, T. D. Dereniak, “Geometrical and trigonometric optics,” in Geometrical and trigonometric optics, Cambridge: Cambridge University Press, 2008.

[15] “Coma (optics),” Wikipedia, 08-Mar-2019. [Online]. Available: https://en.wikipedia.org/wiki/Coma_(optics). [Accessed: 08-Jun-2019].

[16] Artal, Pablo. Handbook of Visual Optics, Two-Volume Set. 2nd ed. CRC Press, 2017.

[17] "Defocus aberration," Wikipedia, 07-Mar-2019. [Online]. Available: https://en.wikipedia.org/wiki/Defocus_aberration. [Accessed: 10-Jun-2019].

[18] R. G. Lane and M. Tallon, "Wave-front reconstruction using a Shack–Hartmann sensor," Applied Optics, vol. 31, no. 32, p. 6902, Nov. 1992.

[19] "Zernike polynomials," Wikipedia, 24-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Zernike_polynomials. [Accessed: 30-May-2019].

[20] V. Lakshminarayanan and A. Fleck, "Zernike polynomials: a guide," Journal of Modern Optics, vol. 58, no. 18, pp. 1678–1678, Oct. 2011.

[21] B. C. Platt and R. Shack, "History and Principles of Shack-Hartmann Wavefront Sensing," Journal of Refractive Surgery, vol. 17, no. 5, 2001.

[22] D. R. Neal, J. Copland, and D. A. Neal, "Shack-Hartmann wavefront sensor precision and accuracy," in Advanced Characterization Techniques for Optical, Semiconductor, and Data Storage Components, 2002.

[23] "Hartmann test," Il Grattavetro. [Online]. Available: https://www.grattavetro.it/hartmann-test/?lang=en. [Accessed: 08-Jun-2019].

[24] Shack-Hartmann Wavefront Sensors. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=5287. [Accessed: 23-Apr-2019].

[25] Panagopoulou, S. I., & Neal, D. R. "Zernike vs. zonal matrix iterative wavefront reconstructor," 2014

[26] S. I. Panagopoulou and D. R. Neal, "Zernike vs. zonal matrix iterative wavefront reconstructor," -, 2005. [Online]. Available: https://www.researchgate.net/publication/268396449_Zernike_vs_Zonal_Matrix_Iterative_Wavefront_Reconstructor

[27] Hartmann Wavefront Analyzer Tutorial. Spiricon, Inc., Logan., Utah., USA. 2004. Available: shorturl.at/ablV7 [Accesed: 08-Apr-2019]

[28] J. M. Bardsley, "Wavefront Reconstruction Methods for Adaptive Optics Systems on Ground-Based Telescopes," SIAM Journal on Matrix Analysis and Applications, vol. 30, no. 1, pp. 67–83, Jan. 2008.

[29] S. Prabhudesai, "Adaptive Optics Reconstruction Methods". Instrumentation Lab. IUCAA, Hawaii, USA. Aug 22, 2011. Available: https://www.ifa.hawaii.edu/Robo-AO/docs/smp_ao_22aug2011_talk.pdf [Accessed: 15-Apr-2019]

[30] W Zou and J. P. Rolland, "Quantifications of error propagation in slope-based wavefront estimations," Journal of the Optical Society of America A, vol. 23, no. 10, p. 2629, Oct. 2006.

[31] R. Navarro, J. Arines, and R. Rivera, "Direct and inverse discrete Zernike transform," Optics Express, vol. 17, no. 26, p. 24269, Dec. 2009.

[32] I. Mochi and K. A. Goldberg, "Modal wavefront reconstruction from its gradient," Applied Optics, vol. 54, no. 12, p. 3780, Apr. 2015.

[33] F. Dai, F. Tang, X. Wang, O. Sasaki, and P. Feng, "Modal wavefront reconstruction based on Zernike polynomials for lateral shearing interferometry: comparisons of existing algorithms," Applied Optics, vol. 51, no. 21, p. 5028, Jul. 2012.

[34] X. Ma, C. Rao, and H. Zheng, “Error analysis of CCD-based point source centroid computation under the background light,” Optics Express, vol. 17, no. 10, p. 8525, May 2009.

[35] S. Thomas, “Optimized centroid computing in a Shack-Hartmann sensor,” in Advancements in Adaptive Optics, 2004.

[36] A. Vyas, M. B., and B. Raghavendra, “Advanced Methods for Improving the Efficiency of a Shack Hartmann Wavefront Sensor,” in Topics in Adaptive Optics, InTech, 2012.

[37] Poynee, L A., “Correlation Wave-Front Sensing Algorithms for Shack-Hartmann-Based Adaptive Optics using a Point Source”. United States: N. p., 2003.

[38] X. Li, X. Li, and C. wang, “Improvement of correlation-based centroiding methods for point source Shack–Hartmann wavefront sensor,” Optics Communications, vol. 411, pp. 187–194, Mar. 2018.

[39] P. Bedggood and A. Metha, “Comparison of sorting algorithms to increase the range of Hartmann-Shack aberrometry,” Journal of Biomedical Optics, vol. 15, no. 6, p. 67004, 2010.

[40] G. Yoon, “Wavefront Sensing and Diagnostic Uses,” in Adaptive Optics for Vision Science, John Wiley & Sons, Inc., pp. 63–81, 2005.

[41] J. Pfund, N. Lindlein, and J. Schwider, “Dynamic range expansion of a Shack–Hartmann sensor by use of a modified unwrapping algorithm,” Optics Letters, vol. 23, no. 13, p. 995, Jul. 1998.

[42] C. Leroux and C. Dainty, “A simple and robust method to extend the dynamic range of an aberrometer,” Optics Express, vol. 17, no. 21, p. 19055, Oct. 2009.

Appendix-MATLAB Code

This appendix includes all the code developed in this thesis, corresponding to each of the functions that were mentioned in Chapter 5. Comments can be found to help the reader on understanding the code.

definePupil.m

```
function [input_wf_centered,ref_wf_centered,parameters] =
definePupil(input_wf,ref_wf,parameters)
%%%This functions returns the parts of the image of the reference wavefront
%%%and the input wavefront that corresponds to the pupil definition.

%%Calculate the center of the pupil by averaging the centroid positon
%%detected
figure;
imshow(input_wf);
title('Detected spots and pupil');
hold on;
s1 = regionprops(imbinarize(input_wf),'centroid');
centroids = cat(1, s1.Centroid);
plot(centroids(:,1), centroids(:,2), 'b*');
centroid_global = [mean(centroids(:,1)), mean(centroids(:,2))];
plot(centroid_global(1),centroid_global(2),'b*','Color',[1,0,0]);


%Depending on the method selected, the radius will be defined

if(strcmp(parameters.radius_definition,'auto'))
    cx = sort(centroids(:,1));
    cy = sort(centroids(:,2));
    xdiff = cx(end)-cx(1);
    ydiff = cy(end)-cy(1);
    maxdiff = max(xdiff,ydiff);
    parameters.spots_from_center =
round((maxdiff/parameters.spot_px_sep)/2)+1;
end

if(strcmp(parameters.radius_definition,'user_click_defined'))
    [x1,y1] = getpts;
    [x2,~] = getpts;
    centroid_global = [x1,y1];
    xdif = x2-x1;
    parameters.spots_from_center = round(xdif/parameters.spot_px_sep);
end

if(strcmp(parameters.radius_definition,'user_text_defined'))
    prompt = 'How many spots from center have to be considered?';
    x = input(prompt);
    parameters.spots_from_center = x;
end

%Save the radius (and gridSize) in parameters
parameters.pupil_radius = parameters.spots_from_center *
parameters.spot_px_sep;
parameters.pupil_diameter = parameters.pupil_radius*2;
parameters.gridSize = parameters.spots_from_center*2;
pr = parameters.pupil_radius;


%%Plot the circle corresponding to our pupil definition
th = 0:pi/50:2*pi;
xunit = pr * cos(th) + centroid_global(1);
yunit = pr * sin(th) + centroid_global(2);
```

```
plot(xunit, yunit);

%%Now we can crop the image following the pupil definition
up_lim = centroid_global - pr;
input_wf_centered = imcrop(input_wf,[up_lim(1),up_lim(2),pr*2-1,pr*2-1]);
ref_wf_centered = imcrop(ref_wf,[up_lim(1),up_lim(2),pr*2-1,pr*2-1]);

%%The last step of the function is to define what are the pixels that lay
%%inside the pupil, so that we take them into account for later
%%calculations of centroids and local gradients.
s2 = regionprops(imbinarize(input_wf_centered),'centroid');
centroids = cat(1, s2.Centroid);
pupil_center = [mean(centroids(:,1)), mean(centroids(:,2))];
imageSize = size(input_wf_centered);
ci = [pupil_center(1), pupil_center(2), pr];% center and radius of circle
([c_row, c_col, r])
[xx,yy] = ndgrid((1:imageSize(1))-ci(1),(1:imageSize(2))-ci(2));
mask = double((xx.^2 + yy.^2)<ci(3)^2);
mask(mask==0) = 0;
input_wf_centered = input_wf_centered.*mask;

%%Normalize to maximum
input_wf_centered = input_wf_centered/max(input_wf_centered(:));
ref_wf_centered = ref_wf_centered/max(ref_wf_centered(:));

%Shift the images to correct misaligment between reference and input
ref_wf_centered = shiftFix(input_wf_centered,ref_wf_centered,parameters);

end
```

Listing 2: *definePupil.com*

sortSpots.m

```
function [c_rdata_sorted,c_idata_sorted] =
sortSpots(input_wf_centered,ref_wf_centered,parameters)
%%This function is responsible of sorting the spots, matching reference and
%%displced spots, using the method selected.

%%By default, the spots are sorted by minimum distance. This is done
%%independently of the method selected, as will be used for other sorting
%%methods.
    gridSize = parameters.gridSize;
    %Find the position of the spots
    s1 = regionprops(imbinarize(ref_wf_centered),'centroid');
    c_rdata = cat(1, s1.Centroid);
    s2 = regionprops(imbinarize(input_wf_centered),'centroid');
    c_idata = cat(1, s2.Centroid);
    c_idata_aux = nan(size(c_rdata));
    %Calculate distance matrix
    dist = pdist2(c_rdata,c_idata);
    %Sort by minimum distance
    for i = 1:size(c_idata,1)
       [~,idx] = min(dist(:,i));
       c_idata_aux(idx,:) = c_idata(i,:);
    end
    lim = 1;
    c_rdata_sorted = cell(gridSize,gridSize);
    c_idata_sorted = cell(gridSize,gridSize);
    for i=1:gridSize
        lim = lim + gridSize;
        a = c_rdata(lim-gridSize:lim-1,2);
        [~,idx] = sort(a);
        for j=1:gridSize
            c_rdata_sorted{j,i} = c_rdata(idx(j)+lim-gridSize-1,:);
            c_idata_sorted{j,i} = c_idata_aux(idx(j)+lim-gridSize-1,:);
        end
    end
```

```
    %%If the user has selected the second method, the algorithm will be
    %%executed (Zernike-Based sorting)
    if(parameters.sort_method == 2)

     for i=1:gridSize
        for j=1:gridSize
            if(isnan(c_idata_sorted{j,i}))
                 c_idata_sorted{j,i} =  c_rdata_sorted{j,i};
            end

        end
     end
     %Compute the Zernike Polynomials and prepare the variables
        [~,ZF,unitcircle] =
calculateZernikeFunctions(parameters.zernikeorder,parameters.gridSize);
        refX = cellfun(@(v)v(1),c_rdata_sorted);
        refY= cellfun(@(v)v(2),c_rdata_sorted);
        inputX = cellfun(@(v)v(1),c_idata_sorted);
        inputY = cellfun(@(v)v(2),c_idata_sorted);
        center = size(refX,1)/2;
        gs = parameters.gridSize;
        s = 2;
    %Now the first step of the algorithm is executed.
    for i = 1: gs/2-1
        %Select the corresponding portion of the images of each iteration
        subrx = imcrop(refX,[center-s/2 + 1,center-s/2 + 1,s-1,s-1]);
        subry = imcrop(refY,[center-s/2 + 1,center-s/2 + 1,s-1,s-1]);
        subix = imcrop(inputX,[center-s/2 + 1,center-s/2 + 1,s-1,s-1]);
        subiy = imcrop(inputY,[center-s/2 + 1,center-s/2 + 1,s-1,s-1]);
        sr.X = padarray(subrx-subix,[(gs/2-s/2),(gs/2-s/2)],NaN,'both');
        sr.Y = padarray(subry-subiy,[(gs/2-s/2),(gs/2-s/2)],NaN,'both');
        sr.X(isnan(sr.X)) = 0;
        sr.Y(isnan(sr.Y)) = 0;
        %Find the estimated coefficients for that portion of the image and
        %reconstruct the wavefront
        [C,~] = modalReconstructWF(ZF,sr,unitcircle,parameters);
        WF_i = 0;
        for j = 1:21
            WF_i = WF_i + (ZF{j} *
str2double(C(j,1))).*((parameters.focal_length*parameters.wavelength)/(param
eters.pixel_spacing*parameters.lens_diameter));
        end
        %Compute the local gradients of the wavefront and estimate the
        %tentative positions of the spots of the next area of searching
        [FX,FY] = gradient(WF_i);
        FX(isnan(FX)) = 0;
        FY(isnan(FY)) = 0;
        idx = zeros(s,s);
        idx = padarray(idx,[1,1],1,'both');
        idx = padarray(idx,[(gs/2-s/2)-1,(gs/2-s/2)-1],0,'both');
        idx = logical(idx);
        subFX = FX(idx);
        subFY = FY(idx);
        subrefX = refX(idx);subrefY = refY(idx);
        apx = subrefX - subFX;apy = subrefY - subFY; ap = horzcat(apx,apy);
        %With the estimated positions, match to the real ones by minimum
        %distance
        d = pdist2(ap,c_idata);
        for j = 1: size(ap,1)
            [~,ind] = min(d(j,:));
            ap(j,:) = c_idata(ind,:);
        end
        inputX(idx) = ap(:,1);
        inputY(idx) = ap(:,2);
        s = s+2;%Extend the area of searching
    end
```

```
    %The second step of the algorithm is executed now. First we set the
    %initial variables and the max number of iterations
    maxiter = 100;
    counter = 0;
    running = true;
    while(running)
        %Find the localGradients with the resulting sorted spots of the
        %firts step
        sr.X = refX-inputX;
        sr.Y = refY-inputY;
        sr.X(isnan(sr.X)) = 0;
        sr.Y(isnan(sr.Y)) = 0;
        %Estimate the coefficients and reconstruct the wavefront
        [C,~] = modalReconstructWF(ZF,sr,unitcircle,parameters);
        WF_i = 0;
        for j = 1:21
            WF_i = WF_i + (ZF{j} *
str2double(C(j,1))).*((parameters.focal_length*parameters.wavelength)/(param
eters.pixel_spacing*parameters.lens_diameter));
        end
        %Compute the local gradients and estimate the positions of the
        %spots
        [FX,FY] = gradient(WF_i);
        FX(isnan(FX)) = 0;
        FY(isnan(FY)) = 0;
        subFX = FX(:);
        subFY = FY(:);
        subrefX = refX(:);subrefY = refY(:);subref =
horzcat(subrefX,subrefY);
        apx = subrefX - subFX;apy = subrefY - subFY; ap = horzcat(apx,apy);
        %With the estimated positions, match to the real ones by minimum
        %distance, just as before.
        d = pdist2(ap,c_idata);
        acc = zeros(size(ap,1),1);
        for j = 1: size(ap,1)
            [~,ind] = min(d(j,:));
            acc(j) = ind;
            ap(j,:) = c_idata(ind,:);
        end
        %Look for repeated associations, and eliminate them for the next
        %iteration
        for j = 1: size(ap,1)
            ind = acc(j);
            r = find(acc==ind);
            if(size(r,1)>1)
                ap(r,:) = NaN;
            end
        end

        %If there are no repeated associations, end the loop.
        if(size(ap(isnan(ap)),1) == 0)
            running = false;
        end

        inputX(:) = ap(:,1);
        inputY(:) = ap(:,2);
        %Check if the maximum numner of iterations has been reached to end
        %the loop
        counter = counter +1;
        if(counter == maxiter)
            running = false;
        end
    end
        %Prepare the variables for exiting the function
    refX = reshape(refX,gridSize,gridSize);
    refY = reshape(refY,gridSize,gridSize);
    inputX = reshape(inputX,gridSize,gridSize);
    inputY = reshape(inputY,gridSize,gridSize);
```

```
     for i=1:gridSize
        for j=1:gridSize
            c_rdata_sorted{i,j} = [refX(i,j),refY(i,j)];
            c_idata_sorted{i,j} = [inputX(i,j),inputY(i,j)];
        end
     end

    end
end
```

Listing 3: *sortSpots.m*

calculateLocalGradients.m

```
function localGradients =
calculateLocalGradients(input_wf_centered,ref_wf_centered,c_rdata_sorted,c_i
data_sorted,parameters)
%%This function returns the local gradient at each grid position. To do so,
%%it uses the reference image (calibrated) and compares it with the input.

    gridSize = parameters.gridSize;
    localGradients.X = zeros(gridSize,gridSize);
    localGradients.Y = zeros(gridSize,gridSize);
    ignore_sorting = parameters.ignore_sorting;

if(ignore_sorting)
    winsize = parameters.spot_px_sep;
    %%From each center of the grid calculated above we crop a portion of the
    %%ref image and the input image of size of the sub aperture and find the
    %%centroid for each one. Then we find the localGradients, as it is only
a
    %%substraction.
    for i = 1:gridSize
        for j = 1:gridSize
            center = round(c_rdata_sorted{i,j});
            sub_ref_area = imcrop(ref_wf_centered,[center(1)-
winsize/2,center(2)-winsize/2,winsize-1,winsize-1]);
            sub_i_area = imcrop(input_wf_centered,[center(1)-
winsize/2,center(2)-winsize/2,winsize-1,winsize-1]);
            c_ref =calculateCentroid(sub_ref_area,parameters);
            if(sum(sub_i_area(:))<0.1)
                c_i = [NaN,NaN];
            else
                c_i = calculateCentroid(sub_i_area,parameters);
            end
            localGradients.X(i,j) = c_ref(1)-c_i(1);
            localGradients.Y(i,j) = c_ref(2)-c_i(2);
            c_rdata_sorted{i,j} = center + c_ref - winsize/2-1;
            c_idata_sorted{i,j} = center + c_i - winsize/2-1;
        end
    end
else
    winsize = 10; %This size can be changed to be lower, and should be in
    %those cases of great distortion.

    %%Centroids are calculated independently from each detected spot of the
    %%reference image and distorted image.
    for i = 1:gridSize
        for j = 1:gridSize
            center_ref = round(c_rdata_sorted{i,j});
            center_i = round(c_idata_sorted{i,j});
            sub_ref_area = imcrop(ref_wf_centered,[center_ref(1)-
winsize/2,center_ref(2)-winsize/2,winsize-1,winsize-1]);
            sub_i_area = imcrop(input_wf_centered,[center_i(1)-
winsize/2,center_i(2)-winsize/2,winsize-1,winsize-1]);
            c_ref = calculateCentroid(sub_ref_area,parameters);
            if(sum(sub_i_area(:))<0.1)
                c_i = [NaN,NaN];
            else
                c_i = calculateCentroid(sub_i_area,parameters);
```

```
            end
            c_rdata_sorted{i,j} = center_ref + c_ref - winsize/2-1;
            c_idata_sorted{i,j} = center_i + c_i - winsize/2-1;
            localGradients.X(i,j) = c_rdata_sorted{i,j}(1) -
c_idata_sorted{i,j}(1);
            localGradients.Y(i,j) = c_rdata_sorted{i,j}(2) -
c_idata_sorted{i,j}(2);
        end

    end
end

%%We plot the resulting centroids and gradients
figure;
imshow(ref_wf_centered + input_wf_centered); title('Centroids and slope
measurements');
hold on;
c_rdata_aux.X = cellfun(@(v)v(1),c_rdata_sorted);
c_rdata_aux.Y = cellfun(@(v)v(2),c_rdata_sorted);
c_idata_aux.X = cellfun(@(v)v(1),c_idata_sorted);
c_idata_aux.Y = cellfun(@(v)v(2),c_idata_sorted);
refX = c_rdata_aux.X(:);
refY = c_rdata_aux.Y(:);
inputX = c_idata_aux.X(:);
inputY = c_idata_aux.Y(:);
for i=1:size(refX,1)
        plot(refX(i),refY(i),'b*');
        plot(inputX(i),inputY(i),'b*','Color',[1,0,0]);
        drawArrow([refX(i),refY(i)],[inputX(i),inputY(i)]);
end

%%Prepare localGradients for returning value
localGradients.X = inpaint_nans(localGradients.X);
localGradients.Y = inpaint_nans(localGradients.Y);

end
```

Listing 4: *calculateLocalGradients.m*

calculateCentroid.m

```
function [centroid] = calculateCentroid(subI,parameters)
%%This function is responsible for computing the position of the light spot
%%on a given subimage. The technique specified in parameters will be used.
[l1,l2] = size(subI);
technique = parameters.centroid_technique;
w = parameters.winSize;
inWeight = parameters.inWeight;
WCoG_gaussian_var = parameters.WCoG_var;
IWCoG_it = parameters.IWCoG_it;
size_template = parameters.size_template;
var_template = parameters.var_template;

%% No noise reduction
if(technique == 1)
    centroid = coG(subI);
end

%% Thresholding
if(technique == 2)
    subI = threshold(subI);%Apply thresholding
    centroid = coG(subI);
end

%% Windowing
if(technique == 3)
    %Find maximum intensity pixel
    [~,imax] = max(subI(:));
    [row,col] = ind2sub(size(subI),imax(1));
```

```
    %Apply window
    mask = zeros(size(subI));
    if ~(row-w <1 || col-w <1 || row+w > size(subI,1) || col+w >
size(subI,1))
        mask(row-w:row+w,col-w:col+w) = 1;
        subI(~mask) = 0;
    end
    centroid = coG(subI);
end

%% Weighted CoG by Intensity
if(technique == 4)
    subI = subI.^inWeight;
    centroid = coG(subI);
end

%% Weighted CoG by Gaussian
if(technique == 5)
%Gaussian is centered on the tentative centroid possition with the regular
%method. This can be changed, by selecting the pixel of maximum intensity
%or calculating the tentative position by other method.
centroid = coG(subI);
W = customgauss([l1,l2],WCoG_gaussian_var,...
WCoG_gaussian_var,0,0,1,[-l1/2 + centroid(2),-l2/2 + centroid(1)]);
subI = subI.*W;
end

%% Iterative CoG by Gaussian
if(technique == 6)
    for i = 1:IWCoG_it
        centroid = coG(subI);
        W = customgauss([l1,l2],WCoG_gaussian_var,...
            WCoG_gaussian_var,0,0,1,[-l1/2 + centroid(2),-l2/2 +
centroid(1)]);
        subI = subI.*W;
    end
end

%% Correlation Method
if(technique == 7)
    template = fspecial('gaussian', [size_template size_template],
var_template); %Spot light model
    template = template/max(template(:)); %Normalize to maximum
    correlations = normxcorr2(template,subI);
    [~,maxCorr] = max(abs(correlations(:)));%Get the maximum value
    [row,col] = ind2sub(size(correlations),maxCorr);%Position of the maximum
    %Sub pixel calculation
    deltarow = (log(correlations(row-1,col)/correlations(row+1,col)))/...
    (2*log((correlations(row+1,col)*correlations(row-
1,col))/(correlations(row,col)^2)));
    deltacol = (log(correlations(row,col-1)/correlations(row,col+1)))/...
    (2*log((correlations(row,col+1)*correlations(row,col-
1))/(correlations(row,col)^2)));
    rowAbs = row-size_template + deltarow +size_template/2 + 0.5;
    colAbs = col-size_template + deltacol +size_template/2 + 0.5;
    centroid = [colAbs,rowAbs];
end

%% Adaptive Template Correlation Method
if(technique == 8)
%     [parameters.size_template,~] =
calibrateTemplate(size_template,var_template,subI,'size');
    [~,parameters.var_template] =
calibrateTemplate(parameters.size_template,var_template,subI,'var');
    parameters.centroid_technique = 7;
    centroid = calculateCentroid(subI,parameters);
end
%% Auxiliar Functions
```

```
function [sizeT,varT] = calibrateTemplate(sizeT,varT,subI,target)
    intervalSize = 0;intervalVar = 0;
    if(strcmp(target,'var'))intervalVar = 0.05;end
    if(strcmp(target,'size'))intervalSize = 1;end
    maxCprev = 0;c = 0;
    while(true)
        t = fspecial('gaussian', [sizeT sizeT], varT);
        t = t/max(t(:));
        corrs = normxcorr2(t,subI);
        [maxC,~] = max(corrs(:));
        if(maxC < maxCprev)
            intervalSize = -intervalSize;
            intervalVar = -intervalVar;
            c = c+1;
        end
        if(c>=2)
            varT = varT+intervalVar;
            sizeT = sizeT + intervalSize;
            break;
        end
        maxCprev = maxC;
        varT = varT+intervalVar;
        sizeT = sizeT + intervalSize;
    end

end

function subI = threshold(subI)
    Imax = max(subI(:));
    th = 0.2*Imax;
    subI(subI<th) = 0;
end

function centroid = coG(img)
    L1 = size(img,1);
    L2 = size(img,2);
    sumI = 0;
    sumIx = 0;
    sumIy= 0;
    for o = 1:L1
        for p = 1:L2
            sumIx = sumIx + img(o,p) * p;
            sumIy = sumIy + img(o,p) * o;
            sumI = sumI + img(o,p);
        end
    end
    if sumI == 0
        sumI = NaN;
    end
    centroid = [sumIx/sumI,sumIy/sumI];
end

end
```

Listing 5: *calculateCentroid.m*

zonalReconstructWF.m

```
function [X,Y,WF] = zonalReconstructWF(localGradients,parameters)
%This function computes the wavefront phase from the local slope
%measurements obtained previously. Depending on the input method it will
%use different strategies for getting it.

%% Calculate the mesh of X and Y and set initial variables
L = parameters.pupil_diameter;
```

```
x = -L/2:1:L/2-1;
x = x * parameters.pixel_spacing;
[X,Y] = meshgrid(x);
[m,~] = size(localGradients.X);
X = imresize(X,[m,m]);
Y = imresize(Y,[m,m]);
WF_X = zeros(m,m);
WF_Y = zeros(m,m);
method = parameters.zonal_method;


%% Linear Integration Method
if(method==1)
    for i = 1:m
        for j = 2:m
            WF_X(i,j) = WF_X(i,j-1) + localGradients.X(i,j-1);
        end
    end
    for j = 1:m
        for i = 2:m
            WF_Y(i,j) = WF_Y(i-1,j) + localGradients.Y(i-1,j);
        end
    end


WF = WF_X + WF_Y ;
end

%% Linear Integration Method From center
if(method == 2)
    for i = 1:m
        for j = m/2:-1:1
            WF_X(i,j) = WF_X(i,j+1) - localGradients.X(i,j+1);
        end
        for j = m/2+2:m-1
            WF_X(i,j) = WF_X(i,j-1) + localGradients.X(i,j-1);
        end
    end
    for j = 1:m
        for i = m/2:-1:1
            WF_Y(i,j) = WF_Y(i+1,j) - localGradients.Y(i+1,j);
        end
         for i = m/2+2:m-1
            WF_Y(i,j) = WF_Y(i-1,j) + localGradients.Y(i-1,j);
        end
    end

WF = WF_X + WF_Y ;
end


%% Least Square Method - HUDGIN GEOMETRY

if(method==3)

    deltaX = localGradients.X(:); deltaX(isnan(deltaX)) = 0;
    deltaY = localGradients.Y(:); deltaY(isnan(deltaY)) = 0;

    H1 = zeros(size(deltaX,1),m*m);

    k = 0;
    for i = 1:size(deltaX,1)
    k = k+1;
    if(k < size(deltaX,1))
        H1(i,k) = -1;
        H1(i,k+1) = 1;
    end
    end
```

```
    H2 = zeros(size(deltaX,1),m*m);

    k = 0;
    for i = 1:size(deltaX,1)
    k = k+1;
    if(k+m < size(deltaX,1))
        H2(i,k) = -1;
        H2(i,k+m) = 1;
    end
    end


    H = vertcat(H2,H1);
    H = vertcat(H,ones(1,m*m));
    G = vertcat(deltaX,deltaY);
    G = vertcat(G,0);

    WF = H\G;
    WF = reshape(WF,m,m);
end


%% Least Square Method - SOUTHWELL GEOMETRY

if(method==4)

deltaX = localGradients.X(:); deltaX(isnan(deltaX)) = 0;
deltaY = localGradients.Y(:); deltaY(isnan(deltaY)) = 0;

G = vertcat(deltaY,deltaX);
[H,Cs] = getSouthwellMatrix(m);
WF = solveMatrixSystem(H,Cs*G);
WF = reshape(WF,m,m);
end

%% Method Using external function intgrad2
if(method==5)
    WF = intgrad2(localGradients.X,localGradients.Y);
end

%%Apply normalization factors
WF =
WF./((parameters.focal_length*parameters.wavelength)/(parameters.pixel_spaci
ng*parameters.lens_diameter));
end
```

Listing 6: *zonalReconstructWF.m*

calculateZernikeFunctions.m

```
function [Z,ZernikeF,unitcircle] = calculateZernikeFunctions(order,L)
%%This function computes the Zernike Polynomials over the unit circle for a
%%given order and a given length that matches the wavefront length. The
%%result is given in column vectors in the case of Z, where each column is
%%a polynomial, or in a cell array in the case of ZernikeF. The function
%%also gives the logical matrix unitcircle, relevant for the coefficient
%%calculation and modal reconstruction.

%%Calculation of the indexes N and M from the order given (OSA order)
N = []; M = [];
for n = 0:order
N = [N n*ones(1,n+1)];
M = [M -n:2:n];
end

%%Definition of the unit circle
X = -1:2/(L-1):1;
% X = -0.9:0.9*2/(L-1):0.9;
```

```
[x,y] = meshgrid(X);
[phi,r] = cart2pol(x,y);
f_size = size(r);
ZernikeF = cell(1,size(N,2));
unitcircle = r<=1;
r(~unitcircle) = NaN;
phi(~unitcircle) = NaN;
r = r(:);
phi = phi(:);

%Calculation of the value of each polynomial for each r and phi
    for j=1:size(N,2)
        n = N(j);
        m = M(j);
        ZernikeF{j} = zeros(size(r,1),1);
        radialpol = zeros(size(r,1),1);
        for i = 1:size(r,1)
            radialpol(i) = 0;
            for k=0:(n-abs(m))/2
               e1 = (-1)^k;
               e2 = nchoosek(n-k,k);
               e3 = nchoosek(n-2*k,(n-abs(m))/2 - k);
               e4 = r(i)^(n-2*k);
               radialpol(i) = radialpol(i) + e1*e2*e3*e4;
            end
            if m<0
                ZernikeF{j}(i) = radialpol(i)*sin(abs(m)*phi(i));
            else
                ZernikeF{j}(i) = radialpol(i)*cos(m*phi(i));
            end
        end
        ZernikeF{j} = ZernikeF{j} .* sqrt((1+(m~=0))*(n+1)/pi);
    end

%Construction of the Z matrix (columns are each Zernike Polynomial).
Z = zeros(size(r(unitcircle),1),size(M,2));
for i=1:size(ZernikeF,2)
    ZernikeF{i} = reshape(ZernikeF{i},f_size);
    Z(:,i) = ZernikeF{i}(unitcircle);
end

end
```

Listing 7:*calculateZernikeFunctions.m*

calcualteZernikeCoeffs.m

```
function ZernikeCoeffs = calculateZernikeCoeffs(Z,WF)
%%This function computes the  decomponsition of the wavefront calculated
%%into the Zernike polynomials for a given order. Also the classic name
%%is given for the first 15 Zernike coefficients.

ZernikeCoeffs = solveMatrixSystem(Z,WF,2);
ZernikeCoeffs(1) = 0;
nameCoeffs = ["Piston","Tilt","Tip","Oblique
Astigmatism","Defocus","Vertical astigmatism","Vertical Trefoil","Vertical
Coma","Horizontal Coma","Oblique Trefoil","Oblique Quadrafoil","Oblique
secondary astigmatism","Primary spherical","Vertical secondary
astigmatism","Vertical quadrafoil"];
nameCoeffs = nameCoeffs';
textCoeffs = strings(size(Z,2),1);
for i = 1:size(Z,2)
    if i<=15
        textCoeffs(i) = strcat(num2str(i),"-",nameCoeffs(i));
    end
end
```

```
ZernikeCoeffs = horzcat(ZernikeCoeffs,textCoeffs);
end
```

Listing 8: *calculateZernikeCoeffs.m*

modalReconstructWF.m

```
function [ZernikeCoeffsModal, WF] =
modalReconstructWF(ZernikeFunctions,localGradients,unitcircle,parameters)
%This function calculates the wavefront directly from the slope
%measurements following the strategy for modal reconstruction. We obtein
%directly the coefficients, using matrix inversion and the derivatives of
%the Zernike polynomials.

L = size(unitcircle,1); %size of data
% First we build the column vectors for the gradients
% on x and y that belongs only to the unit circle size
% They will be size Nx1
deltaWX = localGradients.X(unitcircle);
deltaWY = localGradients.Y(unitcircle);
% We must concatenate both vectors in one, size 2Nx1
deltaW = vertcat(deltaWX,deltaWY);
deltaW =
deltaW./((parameters.focal_length*parameters.wavelength)/(parameters.pixel_s
pacing*parameters.lens_diameter));

%We set the vectors for the gradients of the zernike functions
% size NxJ
deltaZX = zeros(size(deltaWX,1),size(ZernikeFunctions,2));
deltaZY = deltaZX;

for i=1:size(ZernikeFunctions,2)
    Zaux = ZernikeFunctions{i};
%     Zaux(isnan(Zaux)) = 0;
% Compute the gradients of the functions
    [FX,FY] = gradient(Zaux);
% Now, every NaN value becomes 0
% This is because previously every value outside the unit circle
% was NaN, so when we compute the gradients, every value of the polynomials
% that is on the edge of the unit circle will result on a NaN value for
% the gradient at its position. Making all values 0, we make sure that the
% % gradients have a correct value on the edges.
    FX(isnan(FX)) = 0;
    FY(isnan(FY)) = 0;
% [FX,FY] = diffSurface(Zaux);

    deltaZX(:,i) = FX(unitcircle);
    deltaZY(:,i) = FY(unitcircle);
end
%Vector of the gradients of the zernike polynomials
deltaZ = vertcat(deltaZX,deltaZY);
% Apply least-square method to obtain the coefficients
ZernikeCoeffsModal = calculateZernikeCoeffs(deltaZ,deltaW);
% Reconstruct wavefront from the coefficients obtained
WF = zeros(L,L);
for i = 1:size(ZernikeFunctions,2)
    WF = WF + ZernikeFunctions{i} * str2double(ZernikeCoeffsModal(i,1));
end
 end
```

Listing 9: *modalReconstructWF.m*

showResults.m

```
function
showresults(img,corX,corY,WF,ZernikeFunctions,ZernikeCoeffs,unitcircle)
%%This function just shows the results obtained

 X = -1:2/(size(WF,1)-1):1;
[x,y] = meshgrid(X);
[PHI,R] = cart2pol(x,y);
WF(~unitcircle) = NaN;

figure;
subplot(2,2,1);
imshow(img);
title('Sensor image');
subplot(2,2,2);
surf(corX,corY,WF,'edgecolor', 'none');
xlabel('X distance in \mum');
ylabel('Y distance in \mum');
zlabel('Z distance in \lambda / mm');
title('Wavefront');
subplot(2,2,[3,4]);
    pos = get(gca, 'Position');
    pos(1) = 0.2;
    pos(3) = 0.7;
    set(gca, 'Position', pos)
barh(double(ZernikeCoeffs(1:15,1)));
title('First 15 Zernike coefficients (\lambda)');
set(gca,'yticklabel',ZernikeCoeffs(1:15,2));

ZernikeFcart = cell(1,size(ZernikeFunctions,2));
for j=1:size(ZernikeFcart,2)
    [X,Y,ZernikeFcart{j}] = pol2cart(PHI,R,ZernikeFunctions{j});
end

mplot('new');
for i = 1:size(ZernikeFunctions,2)
    fun = @(ax) surf(ax,X,Y,ZernikeFcart{i},'edgecolor', 'none');
mplot(fun);
end

end
```

Listing 10: *showResuts.m*

drawArrow.m

```
function drawArrow(p1,p2)
%%Function to draw an arrow on an image
    dp = p2-p1;
    quiver(p1(1),p1(2),dp(1),dp(2),0)
end
```

Listing 11: *drawArrow.m*

getShouthwellMatrix.m

```
function [H,Cs] = getSouthwellMatrix(m)
%%This function returns the matrix modelling the Southwell geometry for a
%%given size of square area m.

 H1 = zeros((m-1)*(m),m^2);
    k = 0;
    c = 0;
    for i = 1:(m-1)*(m)
    k = k+1;
    c = c+1;
    if(c==m)
        c=1;
        k = k+1;
```

```
    end
    H1(i,k) = -1;
    H1(i,k+1) = 1;
    end


    H2 = zeros((m-1)*(m),m^2);
    k = 0;
    for i = 1:1:(m-1)*(m)
    k = k+1;
        H2(i,k) = -1;
        H2(i,k+m) = 1;
    end

Cs1 = 1/2*(abs(H1));Cs1 = horzcat(Cs1,zeros(size(Cs1)));
Cs2 = 1/2*(abs(H2));Cs2 = horzcat(zeros(size(Cs2)),Cs2);
H = vertcat(H2,H1);H = vertcat(H,ones(1,m*m));
Cs = vertcat(Cs2,Cs1);Cs = vertcat(Cs,zeros(1,size(Cs,2)));
H = sparse(H);
Cs = sparse(Cs);
end
```

Listing 12: *getShouthwellMatrix.m*

RMSE.m

```
function RMSE = RMSE( T, Y )
%%This function computes the RMSE between T and Y
 E    = T-Y;
 SQE  = E.^2;
 MSE  = mean(SQE(:));
 RMSE = sqrt(MSE);
end
```

Listing 13: *RMSE.m*

shiftFix.m

```
function [ref_wf_centered] =
shiftFix(input_wf_centered,ref_wf_centered,parameters)
%%This function shifts the input and reference image so that the mismatch
%%introduced by misaligment is vanished.

gridSize = parameters.gridSize;
iterations = 10; %Set number of iterations
figure;
title('Decentred images');
imshow(ref_wf_centered+input_wf_centered);
parameters.sort_method = 1; %Select sorting by minum distance
for k = 1:iterations
    %First sort the spots by minimum distance
    [c_rdata_sorted,c_idata_sorted] =
sortSpots(input_wf_centered,ref_wf_centered,parameters);
    localGradients.X = zeros(gridSize,gridSize);
    localGradients.Y = zeros(gridSize,gridSize);
    %Compute Local Gradients
     for i=1:gridSize
        for j=1:gridSize
            localGradients.X(j,i) = c_rdata_sorted{j,i}(1) -
c_idata_sorted{j,i}(1);
            localGradients.Y(j,i) = c_rdata_sorted{j,i}(2) -
c_idata_sorted{j,i}(2);
        end
     end
    %Calculate the averege shift
    localGradients.X(isnan(localGradients.X)) = 0;
    localGradients.Y(isnan(localGradients.Y)) = 0;
    shiftX = round(mean(localGradients.X(:)));
    shiftY = round(mean(localGradients.Y(:)));
```

```
    %Shift the images by that calculated average shift
    ref_wf_centered = imtranslate(ref_wf_centered,[-shiftX, -shiftY]);
end

end
```

Listing 14: *shiftFix.m*

solveMatrixSystem.m

```
function [X] = solveMatrixSystem(A,B,method)
%%This functions solves a matrix system (AX = B) by using the specified
%%method
if nargin < 3
    method = 2;
end

if(method == 1)
    X = mldivide(A,B);
end

if(method == 2)
   X = lsqminnorm(A,B);
end

if(method == 3)
    X = pinv(A)*B;
end

end
```

Listing 15: *solveMatrixSystem.m*

getDistortedWavefront.m

```
function [WF_ref,C_ref,idist,iref] =
getDistortedWavefront(Z,ZernikeFunctions,unitcircle,parameters,C)
%%This function calculates a random wavefront from certain Zernike
%%coefficients randomly generated. It will return the wavefront, the
%%coefficients, the image from the sensor corresponding to the generated
%%wavefront and the reference image.

%% Generation of the wavefront
if(isstring(C))
    if strcmp("random",C)
        C_ref = rand(size(Z,2),1)*0.8; C_ref(1) = 0;
        C_ref(:) = 0; C_ref(13) = 0.8;
    end
else
    C_ref = C;
end
L = parameters.spot_max_from_center*2;
WF_ref = zeros(L,L);
for i = 1:size(Z,2)
    WF_ref = WF_ref + C_ref(i)*ZernikeFunctions{i};
end
% WF_ref(isnan(WF_ref)) = 0;
%% Generation of the image corresponding to the wavefront
%Compute partial derivatives of the WF
[FX,FY] = gradient(WF_ref);
FX(isnan(FX)) = 0;
FY(isnan(FY)) = 0;

% Find the difference in pixels modelling the sensor
localGradients.X=
FX.*((parameters.focal_length*parameters.wavelength)/(parameters.pixel_spaci
ng*parameters.lens_diameter));
```

```
localGradients.Y =
FY.*((parameters.focal_length*parameters.wavelength)/(parameters.pixel_spaci
ng*parameters.lens_diameter));

%Calculate the location of the spots
I = zeros(L*parameters.spot_px_sep);
centroids_ref = getRefCentroids(I,parameters.spot_px_sep);
centroids_dist.X = centroids_ref.X - localGradients.X;
centroids_dist.Y = centroids_ref.Y - localGradients.Y;

%Generate reference and distorted image
idist = placeLightSpots(centroids_dist,I);
iref = placeLightSpots(centroids_ref,I);

% [c_rdata_sorted,c_idata_sorted] = sortSpots(idist,iref,parameters);
% parameters.centroid_technique = 1;
% localGradients =
calculateLocalGradients(idist,iref,c_rdata_sorted,c_idata_sorted,parameters)
;
% [C_ref,WF_ref] =
modalReconstructWF(ZernikeFunctions,localGradients,unitcircle,parameters);

%% Auxiliary Functions

%%In order to place the light spots, there are two approaches. The first
%%version of the function works in general, but it takes longer time
%%because it needs to calculate for each spot the gaussian over the full
%%image. The second version is much faster, but it only works when the spot
%%is not out of the sub-area of the grid. That means that for high
%%distortion the second version can't be used because it will fail.
function I = placeLightSpots(centroids,I)
    for k=1:size(centroids.X,1)
        for l=1:size(centroids.X,2)
            center = [-size(I,1)/2 + centroids.Y(k,l),-size(I,1)/2 +
centroids.X(k,l)];
            light_spot = customgauss(size(I),1.5,1.5,0,0,1,center);
            I = I + light_spot;
        end
    end
end

%Place the light spots in their corresponding spots
% function I = placeLightSpots(centroids,I)
%     figure;
%     jump = parameters.spot_px_sep;
%     limY = -jump + 1;
%     for k=1:size(centroids.X,1)
%         limY = limY + jump;
%         limX = -jump + 1;
%         for l=1:size(centroids.X,2)
%             limX = limX + jump;
%             center = [centroids.Y(k,l)-limY-(jump/2) + 1,centroids.X(k,l)-
limX-(jump/2) + 1];
%             light_spot = customgauss([jump,jump],1.5,1.5,0,0,1,center);
%             I(limY:limY+jump-1,limX:limX+jump-1) = light_spot;
%
%             imshow(I);
%         end
%     end
% %     I(I<0.001) = 0;
% end


%Compute the positions of the reference spots
function centroids = getRefCentroids(img,gridSize)
[m,n] = size(img);
centroids.X = zeros(m/gridSize,n/gridSize);
centroids.Y = zeros(m/gridSize,n/gridSize);
```

```
    for i = 1:size(centroids.X,1)
        for j = 1:size(centroids.X,2)
            sumcorX = 0;
            sumcorY = 0;
           for y = 1:gridSize
             corY = y + (i-1)*gridSize;
             sumcorY = sumcorY + corY;
           end
           for x = 1:gridSize
              corX = x + (j-1)*gridSize;
              sumcorX = sumcorX + corX;
           end
           centroids.X(i,j) = (sumcorX/gridSize);
           centroids.Y(i,j) = (sumcorY/gridSize);
        end
    end
end

end
```

Listing 16: *getDistortedWavefront.m*

testNoiseReductionTechnique.m

```
function RMS_WF =
testNoiseReductionTechnique(parameters,technique,snr,meanV)

%%This function test the technique for noise reduction that is selected by
%%the input "technique" and gives back the RMSE for the wavefront generated

%Set the technique used
parameters.centroid_technique = technique;
%Compute the Zernike Polynomials
[Z,ZernikeFunctions,unitcircle] =
calculateZernikeFunctions(parameters.zernikeorder,parameters.gridSize);
%Generate a random distorted wavefront
[WF_ref,ZC_ref,idist,iref] =
getDistortedWavefront(Z,ZernikeFunctions,unitcircle,parameters,"random");
%From SNR calculate the variance of the error
varnoise = var(idist(:))/snr;
%Add Gaussian error of specific variance
idist = imnoise(idist,'gaussian',meanV,varnoise);
%Reconstruct the wavefront with the noisy image and find the RMSE between
%reference and estimated wavefront
[c_rdata_sorted,c_idata_sorted] = sortSpots(idist,iref,parameters);
localGradients =
calculateLocalGradients(idist,iref,c_rdata_sorted,c_idata_sorted,parameters)
;
[ZernikeCoeffsModal,WFmodal] =
modalReconstructWF(ZernikeFunctions,localGradients,unitcircle,parameters);
WFmodal(isnan(WFmodal)) = 0;
WF_ref(isnan(WF_ref)) = 0;
RMS_WF(1) = RMSE(WF_ref,WFmodal);
end
```

Listing 17: *testNoiseReductionTechnique.m*

testRangeExtensionTechnique.m

```
function [RMS_WF,pass,t] = testRangeExtensionTechnique(C,parameters)
%%This function is dedicated to test a given range extension technique. For
%%that a vector of Zernike coefficients C is given as an input to generate
%%the random wavefront.

%Compute Zernike Functions
[Z,ZF,unitcircle] =
calculateZernikeFunctions(parameters.zernikeorder,parameters.spot_max_from_c
enter*2);
%Generate a wavefront for the specified coefficients
```

```
[WF_ref,C_ref,idist,iref] =
getDistortedWavefront(Z,ZF,unitcircle,parameters,C);
tic
%Apply the sorting technique under test
[c_rdata_sorted,c_idata_sorted] = sortSpots(idist,iref,parameters);
t = toc;
%Reconstruct the wavefront and find if the technique has performed
succesfully
localGradients =
calculateLocalGradients(idist,iref,c_rdata_sorted,c_idata_sorted,parameters)
;
[ZernikeCoeffsModal,WFmodal] =
modalReconstructWF(ZF,localGradients,unitcircle,parameters);
WFmodal(isnan(WFmodal)) = 0;
WF_ref(isnan(WF_ref)) = 0;
RMS_WF(1) = RMSE(WF_ref,WFmodal);
refX = cellfun(@(v)v(1),c_rdata_sorted);
refY= cellfun(@(v)v(2),c_rdata_sorted);
rauxX = refX(:);rauxY = refY(:);raux = horzcat(rauxX,rauxY);
inputX = cellfun(@(v)v(1),c_idata_sorted);
inputY = cellfun(@(v)v(2),c_idata_sorted);

if(isnan(sum(inputX(:))))
    pass = false;
else
    pass = true;
end

%Show the result of the sorting algorithm
figure;
img = idist+iref;
img(parameters.spot_px_sep:parameters.spot_px_sep:end,:,:) = 255;        %#
Change every tenth row to black
img(:,parameters.spot_px_sep:parameters.spot_px_sep:end,:) = 255;        %#
Change every tenth column to black
imshow(img);hold on;
refX = refX(:);
refY = refY(:);
inputX = inputX(:);
inputY = inputY(:);
for i=1:size(refX,1)
        plot(refX(i),refY(i),'b*');
        plot(inputX(i),inputY(i),'b*','Color',[1,0,0]);
        drawArrow([refX(i),refY(i)],[inputX(i),inputY(i)]);
end

end
```

Listing 18: *testRangeExtensionTechnique.m*